AD NUMBER: AD0912452

LIMITATION CHANGES

TO:

Approved for public release; distribution is unlimited.

FROM:

Distribution limited to U. S. Government Agencies only; Test and Evaluation; dated 27 January 1972. Other requests for this document must be referred to U. S. Army Advanced Ballistic Missile Defense Agency, ATTN: RDMH-D, P0 Box 1500, Huntsville, AL 35807.

AUTHORITY

ABMDA ltr dtd 5 Feb 1974

Final Report

SENSOR NETTING PROGRAM

Volume II: Field Demonstrations

By: E. T. BRANDON H. A. OLENDER D. G. FALCONER

Prepared for:

DIRECTOR
U.S. ARMY ADVANCE BALLISTIC MISSILE DEFENSE AGENCY
P.O. BOX 1500
HUNTSVILLE, ALABAMA 35807
Attention: RDMH-C

CONTRACT DAHC60-70-C-0016

STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 · U.S.A.

STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 · U.S.A.

Final Report *May 1973*

SENSOR NETTING PROGRAM

Volume II: Field Demonstrations

By: E. T. BRANDON H. A. OLENDER D. G. FALCONER

Prepared for:

DIRECTOR
U.S. ARMY ADVANCE BALLISTIC MISSILE DEFENSE AGENCY
P.O. BOX 1500
HUNTSVILLE, ALABAMA 35807
Attention: RDMH-C

CONTRACT DAHC60-70-C-0016

SRI Project 1853

Approved by:

P. K. WHALEN, *Director*
System Evaluation Department

E. J. MOORE, *Executive Director*
Engineering Division

Copy No. 14

ABSTRACT

For sensor netting in ballistic missile defense (BMD), data must be transferred from one sensor to another. The purpose of the field demonstrations reported here was to demonstrate specific netting functions entailing data transfer from one radar to another remotely located radar under field conditions. At the White Sands Missile Range, target data from various objects were transferred from AMRAD, a mechanical tracking radar, to HAPDAR, a phased array radar driven by an IBM 360/65 computer using special BMD software. The two radars were separated by about ten miles.

This report describes the software developed for these demonstrations and the tests that were carried out to demonstrate target transfer from one radar to another and target track association or correlation between independent tracks of the two radars.

CONTENTS

ILLUSTRATIONS

TABLES

GLOSSARY

ABMDA	U.S. Army Advanced Ballistic Missile Defense Agency
A/D	analog-to-digital
AMRAD	ARPA measurement radar
AGC	automatic gain control
BMD	ballistic missile defense
BMDOS	BMD operating system
CPU	central processing unit
DRTS	digital recording and timing system
GMT	Greenwich mean time
HAPDAR	Hard Point Demonstration Array Radar
ICBM	intercontinental ballistic missile
IRIG	Inter-Range Instrumentation Group
I/O	input and output
I P	AMRAD's isolated pulse
LSB	least significant bit
MDS	minimum discernible signal
MIPS	millions of machine instructions per second
MSB	most significant bit
MSL	mean sea level
OTDS	object track data set
PAS	Precision Acquisition System
PDA	parallel data adapter
PHSD	preliminary hardsite defense
pps	pulses per second
PREC	AMRAD's precursor pulse

RCA	(Radio Corporation of America)
RCI	radar-computer interface
RCS	radar cross section
RDT	range data terminal
RRI	Riverside Research Institute
RSDS	AMRAD data set
RTI	range time intensity display
SNR	signal-to-noise ratio
SRI	Stanford Research Institute
TPDS	track pointer's data set
TROQ	track return queue
WSMR	White Sands Missile Range

ACKNOWLEDGMENTS

The authors wish to acknowledge the help of personnel of several U.S. Army agencies and their contractors in assisting SRI in accomplishing the field demonstrations at WSMR (White Sands Missile Range).

Mr. E. Nordell and Mr. L. Chavez of SURC, acting as ABMDA's WSMR TESCO (Test and Evaluation and Support Coordination Office), helped us in test planning and coordination of range facilities, and acted as test director for the demonstrations.

The AMRAD/RRI personnel under the direction of Mr. M. Arm assisted us by modifying the hardware at AMRAD for the program and aided us in planning and accomplishing the demonstrations. Special thanks are due Mr. R. Kind for the engineering effort in designing the AMRAD/HAPDAR interface equipment.

The HAPDAR personnel, under the direction of Mr. D. Wright, Sperry Gyroscope Division of Sperry Rand Corporation, helped us by operating the radar and collecting data for check-out and demonstration.

The RCA personnel of the ABMDA Field Test Facility under the direction of Dr. W. C. King helped us by integrating the SRI software into the post-PHSD software. Special thanks are due Dr. K. M. Kessler, a subcontractor from Systems Control, Inc., who coordinated the software integration, and Mr. T. R. Cottler for programming the software interfaces.

Finally, Mr. R. Zientek of National Range Programs Office helped us by providing range information and obtaining WSMR range services necessary for the demonstrations.

I INTRODUCTION

In previous netting studies[1]* the importance of demonstrating in real time the transfer of target data from one radar to another and the correlation of that data with the radar data already in the track file of the other radar was established. Any sensor netting scheme must transfer data from one sensor to another. Also, when a second sensor receives data from a first sensor, it is necessary for the second sensor to determine if the target data it has received is associated with objects already in its track data file. If it is, the data might be used to lessen the uncertainty of the target characteristics. If it is not, a new track should be established on the target.

It is important that an ABM system efficiently process data so as to conserve both the time and the resources devoted to the system's various functions. For target correlation, the track file should be rapidly searched for any target associations. For target handover, no time should be lost in establishing a track on a new target designated to be threatening by another sensor. That is, as few radar pulses as possible should be used to acquire the target and place it into track.

Last year, we demonstrated target handover in real time from one radar to another.[2] Because of computer limitations, we could not demonstrate correlation in real time. We did develop a real-time revision of the correlation algorithm and test its performance with real data.

---

*References are listed at the end of this volume.

In the follow-up project reported here, we demonstrated handover and correlation in real time using the recently installed IBM 360/65 computer driving the HAPDAR radar. Target data was supplied by the AMRAD radar, a mechanically slewed, tracking radar, located about ten miles from HAPDAR on the White Sands Missile Range.

During this period we planned to demonstrate the transfer of a limited search sector from one radar to another. However, schedule slippage at WSMR not under SRI's control precluded accomplishing this task during the contract period.

A summary of the work and our conclusions from the field demonstrations are presented in the next chapter. The approach to the field demonstration is then given, including a description of the hardware and software used for the tests. Chapter IV describes the mission that was used to demonstrate handover and correlation--the sounding rocket Rome-03. Chapter V describes the analysis of the data from that mission.

II SUMMARY AND CONCLUSIONS

A. Introduction

For several years SRI has studied the advantages and benefits of netting radars in ballistic missile defense. Out of these studies a series of field tests were proposed to validate the netting concepts that showed promise. Target handover, defined as one radar transferring track information to another radar to enable that radar to initiate target tracking, is basic to most netting concepts. Target correlation, defined as comparing the radar data on a target transferred from one radar to another radar with the track file of the latter radar to assess whether the target is already in that track file or not, is also an important function for most netting concepts.

Another concept that is normally assumed in netting studies is the transfer of an arbitrary search sector from one radar to another. For example, a fire ball "blacks out" one radar, which then calls in another radar to search behind the fire ball to detect any threatening objects.

The ABMDA Field Test Facility located at WSMR, along with the AMRAD radar located ten miles from the ABMDA Field Test Facility, provided a useful simulation of netted hard site defense radars. The HAPDAR radar and the IBM 360/65 computer at HAPDAR provided a model of a BMD phased-array radar driven by a computer with BMD software. AMRAD, a mechanically scanned tracking radar, simulated another BMD radar by sending target track data to HAPDAR from both its primary tracker and its secondary range-only tracker.

SRI developed the software package, AMROHAP, to scale and format the AMRAD radar data to transmit it to HAPDAR over land lines. SRI also

developed the software package, TRANFT, to smooth the data at the IBM 360/65 computer to form messages capable of initiating tracks on the same targets AMRAD was tracking.

In addition, SRI developed two other software packages for installation at the HAPDAR IBM 360/65. The first, QUECOR, used the handover messages from TRANFT and the HAPDAR track file, OTDS, to determine if any of the tracks already in the track file were associated with the current handover message. This association process is called target correlation.

The second software package, SSERCH, used the AMRAD data smoothed by TRANFT to calculate the beam pointing positions to search the AMRAD designated search sector with the HAPDAR radar. The search data base was planned to be updated periodically by using new data from AMRAD.

A set of test plans was prepared to calibrate the handover system, to test the operations of handover and correlation on ballistic targets of opportunity, and to test correlation and special search on request on radar calibration spheres dropped from high flying aircraft. RCA, the software contractor at the ABMDA Field Test Facility, had schedule slippages in integrating the BMD software with the HAPDAR radar. Because of these delays, it was not possible to carry out the complete set of test plans. However, some calibration data were collected and analyzed, and handover and correlation were demonstrated on a sounding rocket, Rome-03.

B. Software Development

1. AMTOHAP

AMTOHAP takes data from the primary and secondary AMRAD trackers and sends it to the IBM 360/65 computer at HAPDAR via the MILGO interface and kineplex modems. The program, which is resident in the SDS-920 computer at AMRAD, is an intercept driven algorithm that provides the software connection between the AMRAD and HAPDAR radars.

This program was originally developed under a previous contract[2] for ABMDA. For the current contract the program was modified by eliminating unnecessary routines and altering the program to accept both the primary and secondary trackers' coordinates.

Each 120-bit message from this program contains the track identification; the track status; the azimuth, elevation, and range of the target; the signal level of the target; and the time of day the message was valid. A message on each track was sent every 0.1 second for a total message rate of 20 per second.

2. TRANFT

TRANFT takes the data received from AMRAD, converts it to HAPDAR coordinates, smooths it with a tracking filter, using an effective two pulse smoothing, and then gives the fitted data to the HAPDAR operating system. The algorithm requires about eight milliseconds to complete one cycle on the IBM 360/65, a nominal one MIP machine. The total handover delay is about 70 milliseconds, 50 milliseconds for transmission of the message from AMRAD to HAPDAR and 20 milliseconds to compute and order a radar transmission from HAPDAR.

3. QUECOR

The correlation algorithm, QUECOR, determines on request whether an object (or objects) being tracked by one radar is also being tracked by a second radar. It does this operation by comparing a given track from one radar with the track file of a second radar.

QUECOR, installed in the IBM 360/65 computer, can compare three tracks in the HAPDAR track file with one track from TRANFT in five milliseconds. This figure is for closely spaced tracks. With coarse

screening and prescreening to eliminate all but closely spaced tracks, the speed of operation can be increased substantially for typical track spacings.

4. SSERCH

SSERCH is a real time, field test algorithm that generates a series of beam pointing directions and range gates required for HAPDAR to search a volume on special request. This might be the shadowed volume behind a nuclear fireball that temporarily blocks out part of another radar's search volume (in our case AMRAD).

SSERCH was installed in the tactical function library in the IBM 360/65 at the HAPDAR facility. SSERCH was never integrated into the PHSD system due to the priority on the handover and correlation software integration and testing and subsequent delays in achieving that task. Thus, SSERCH has never been validated in real time operation.

C. Field Demonstrations

Four sets of useful tracking data were collected during the contract period. The first, a balloon-borne sphere, was launched about 20 km due north of HAPDAR; it drifted southeast as it climbed in altitude. About 116 seconds of recorded data was obtained. The rest of the data was collected during the Rome-03 mission. The premission balloon-borne sphere was recorded for 73 seconds at an average range of 57 km. The Rome rocket was recorded for eight seconds at an elevation of 70° and a range of 170 km. The Rome parachute-borne payload was tracked for 55 seconds at an average range of 93 km.

1. Calibration

The sphere-drop aircraft were not available for a detailed calibration for the handover experiment; however, the initial balloon-borne sphere was used for first order calibration. The data on this mission indicated a HAPDAR-relative-to-AMRAD range bias of 23 m short, an angle bias of 2.6 millisines above, and an angle bias of 1.8 millisines left.*

Because the range bias was the most critical, this bias was corrected for the ROME mission. However, the Rome-03 mission showed that a bias still remained; it had a slope of 0.49 m per km range, passing through zero bias at 58 km. So far we have no explanation for this linearly varying bias.

2. Handover Results

On the Rome-03 premission sphere, out of 76 attempts, about 68 percent of the handovers resulted in first pulse acquisition and over 98 percent in two pulse acquisition. The first pulse misses can be explained by a low setting of the AGC. Because of the large clutter on the range, RCA did not want to operate the HAPDAR receivers at high gain, creating the possibility of too many hits in the secondary tracker channel. After the first miss, the receiver gain was increased 3 dB, resulting in nearly all of the attempts at handover being successful in two attempts.

Handover on the other targets was successful in some instances, but often the signal-to-noise ratios were too low to expect successful handover or high clutter levels masked handover.

---

*One millisine is one one-thousandth of a sine. Near boresight this is approximately equal to one milliradian.

3. Correlation Results

Track data collected on the Rome-03 premission sphere provides a good demonstration of the operation of the correlation algorithms in real time. During the Rome-03 mission there was a 7.5 meters range bias between AMRAD's first and second tracker. Consequently, although only one target was available, the AMRAD data appeared as if two targets were being handed over. The correlation measure, Q, was sensitive to this small range separation. Even with uncalibrated angle biases remaining and this small range bias present, a decision threshold could be selected to result in correct correlation decisions almost 90% of the time.

For the Rome rocket and payload, no independent tracks were established by HAPDAR. But, some track data was available from the two dedicated HAPDAR tracks. These tracks were refreshed at an interval of every two seconds. As the new handover message is generated it is compared with the old dedicated track to determine the degree of correlation. Since the old dedicated track is many pulses old, it is nearly independent of its initial values. This produced tests of the correlation algorithm at low signal-to-noise ratios.

The Rome rocket data was unique, containing data on two independent targets. The first tracker tracked the rocket booster while the second tracker tracked the payload. These two targets were separated by about 600 meters. Excellent discrimination between these two targets was demonstrated.

From these tests, we can expect that separations of several hundred meters between objects will result in proper correlation decisions with a high degree of success.

D. Implications for HARDSITE Defense

In our analytic studies of radar netting, we have shown that shared search, where several radars in a defense module share the task of searching the threat corridor, can result in an increase in defense system performance. To accomplish this netting technique,. target handover and target correlation are crucial to implementing the operation.

We have demonstrated that target handover and correlation can be performed in a satisfactory manner on slow speed targets in real time. Thus, from these demonstrations, it appears that the value of shared search can indeed be realized.

The next step, a difficult one to be sure, is to demonstrate netted operations with ICBM targets where clutter due to tank break up, penetration aids, and ionized wakes from reentry objects places a severe load on defense resources.

III APPROACH TO FIELD DEMONSTRATIONS

A. Introduction

The objective of the field experiments was to demonstrate, in real time, target data handover from AMRAD (a coherent, monopulse, mechanical tracking radar) to HAPDAR (a multifunction phased array radar). Both of these L-band radars are located in the southeastern section of WSMR. The radars are separated by a distance of about ten miles, as shown in Figure 1. Also shown in the figure is the track of a typical ATHENA mission launched from Green River, Utah; it approaches on an azimuth nearly perpendicular to the base line between the two radars.

The geometric configuration of the two radars approximates the spacings common for HARDSITE defense radars in a defense module. This configuration permits the simulation of data transfer from one radar to another in a defense module and control of the phased array radar based on that data.

Two types of operations common to HARDSITE defense were planned. The first was precision track handover from one radar to another. Here AMRAD tracks a target and then transfers the raw track data to HAPDAR via a data communications link. The data is processed at HAPDAR to generate a precision state vector in HAPDAR coordinates, which includes the target position, velocity, and signal strength. The state vector is first compared with all of the other state vectors in the track file to determine whether or not the handed-over track is already in HAPDAR's track file. After this comparison is made, the handed-over state is updated with radar measurements, initiating a track on the object. This process is called target handover and correlation.

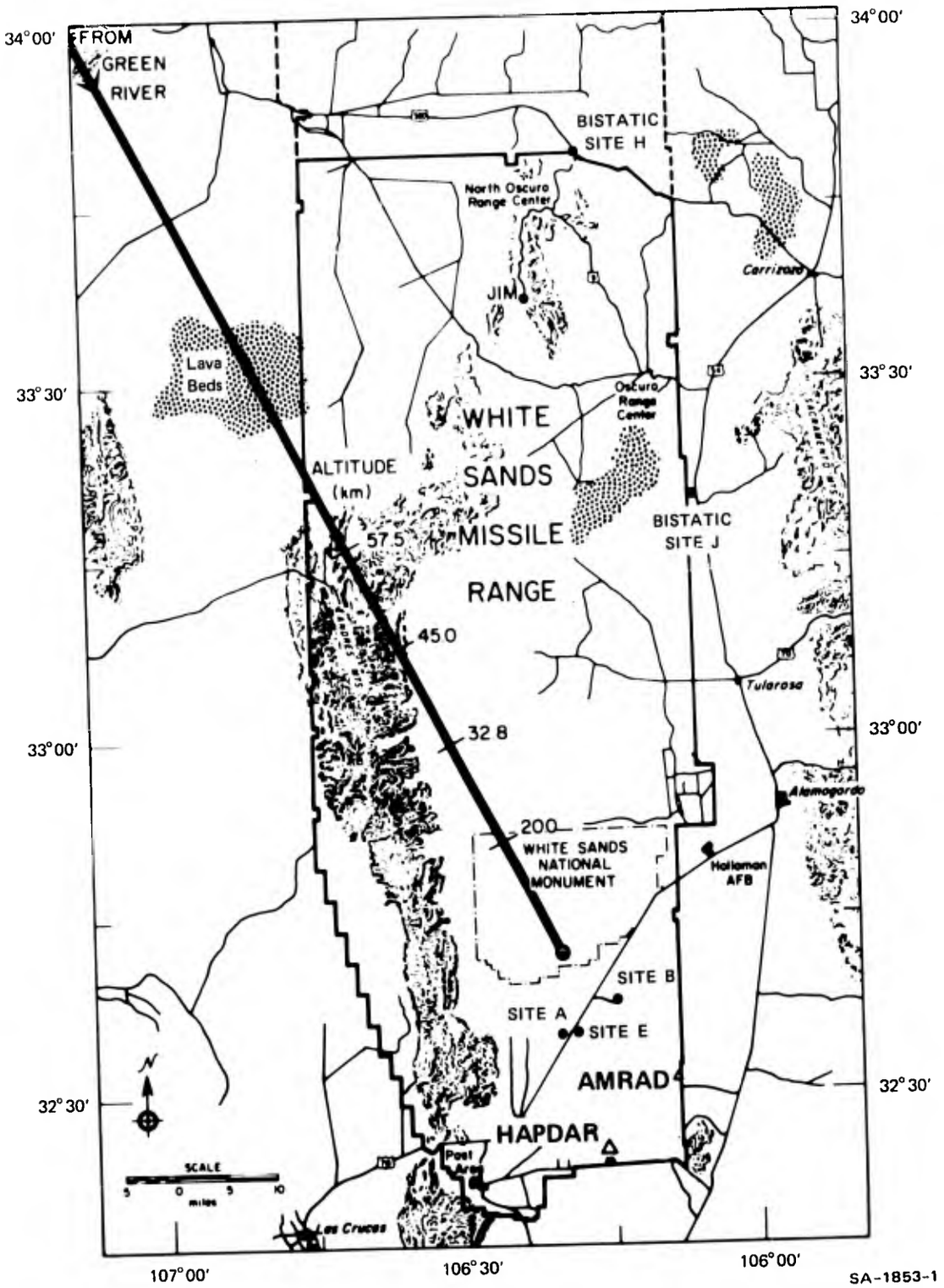


FIGURE 1 MAP OF WSMR SHOWING AMRAD, HAPDAR, AND A TYPICAL HIGH-REENTRY-ANGLE ATHENA TRAJECTORY

The second operation that was planned was search sector handover. Here one radar designates a limited region to be searched for targets by the other radar. The operation as modeled was this: one radar, AMRAD, transferred a sector, one beamwidth wide and 20 km in range, to HAPDAR for search coverage. After receiving the search request, the data was processed at HAPDAR to generate the special search data file consisting of beam positions and range limits for the search. These search orders were then transmitted by the radar, and any detections processed through verification to track.

The handover system block diagram is shown in Figure 2. Data on two radar targets are collected by the AMRAD radar. One is the data from the

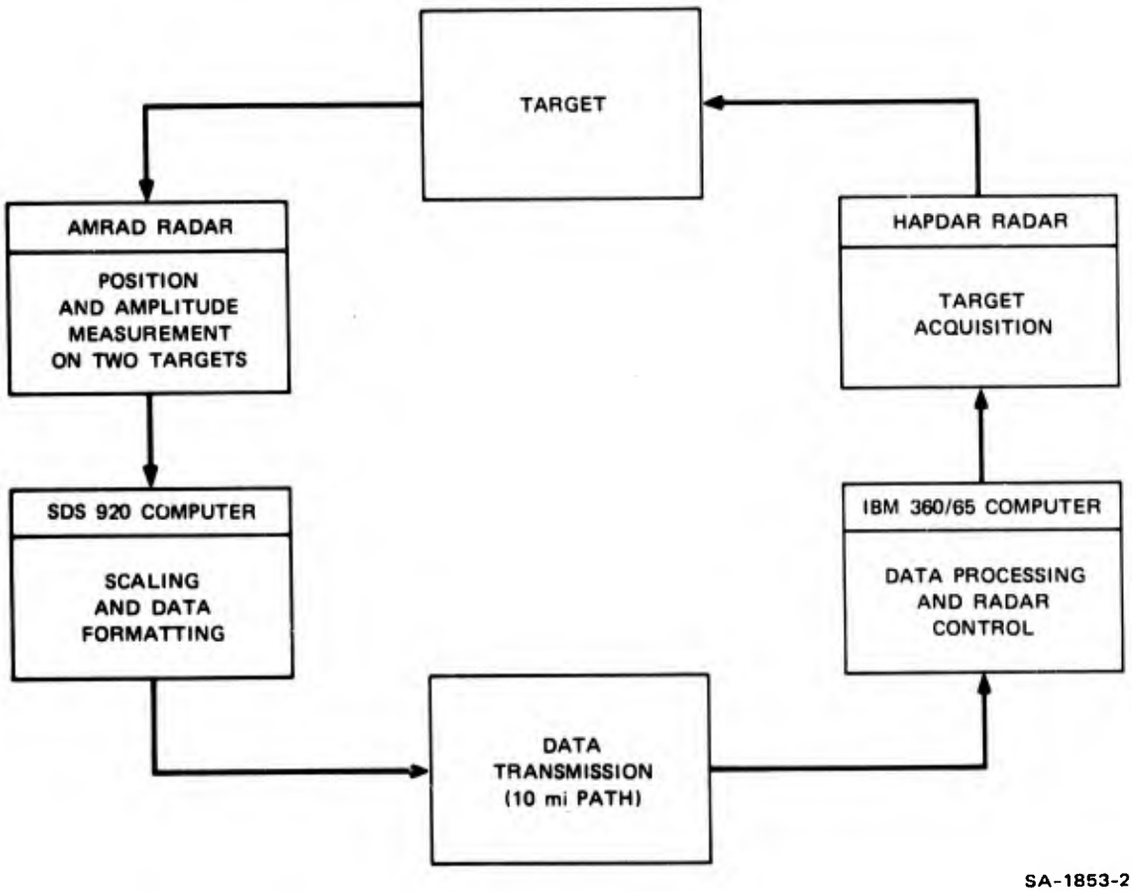


FIGURE 2 HANDOVER SYSTEM BLOCK DIAGRAM

primary radar tracker and the other is the data from the secondary rar e-only tracker. These data are scaled and placed in the proper form to be loaded into the kineplex data modem by the on-site SDS 920 computer.

The data modem transmits the data over dedicated land . s at a rate of 2400 bits/sec. Each track data point (azimuth, elevation, range, and signal strength) requires 120 bits. Accordingly, data from each tracker is sent at ten points per second.

At HAPDAR, the data is demodulated in another kineplex modem. The data is then transferred via several pieces of equipment--the range data terminal (RDT), the radar computer interface (RCI), and the IBM 2701 communications controller--to an IBM 360/65 computer in the HAPDAR facility.

This IBM 360/65 was programmed with special IBM developed, ballistic missile defense software. This software, known as IBM post-PHSD, made possible the real-time control of the HAPDAR radar. SRI developed additional software to filter the incoming AMRAD data, generate the handover state vector, and correlate this vector with the HAPDAR target file. SRI also developed the software for generating a special search data file from AMRAD data.

B. AMRAD

The AMRAD Measurement Program is an experimental investigation of radar observables of reentry objects. It is conducted by RRI (the Riverside Research Institute) under the sponsorship of the WSMR. RRI is assisted in radar site operations by the Raytheon Co. The observations are made using a coherent L-band, monopulse, tracking radar with a high degree of flexibility in the waveforms it can generate.

However, only the basic track data were used in the handover experiments. The characteristics of the radar are listed in Table 1.

Table 1

AMRAD RADAR CHARACTERISTICS

Frequency	1300 MHz
Peak power	10 MW
PRF	50 per second
Antenna gain	44.7 dB
Beamwidth	0.9 degrees
System effective noise temperature	200° K
Compressed pulse width	0.08 μs
S/N on 0.1 m^2 target at 100 km	37 dB
Range accuracy	15 meters
Azimuth accuracy	0.3 mrad
Elevation accuracy	0.3 mrad

The Digital Recording and Timing System (DRTS) at AMRAD is the central timing source of the radar complex and collects, records, and processes data taken on radar targets. It is organized around a general purpose digital computer, the SDS Sigma 5, which controls the generation of the transmitted waveform, the timing of the samples of the received echoes, and the transmission of the data to the recorders.

Radar data under control of the DRTS is supplied to another digital computer, the SDS 920, via the MILGO interface equipment. A block diagram of the AMRAD/HAPDAR interface equipment is shown in Figure 3. Further details are presented in Appendix F. Signal amplitude, azimuth, elevation, range, and time of day (IRIG time) on two targets are supplied to the MILGO interface equipment. In addition, synchronization signals that control the transmission of the radar pulses are transmitted to the MILGO equipment.

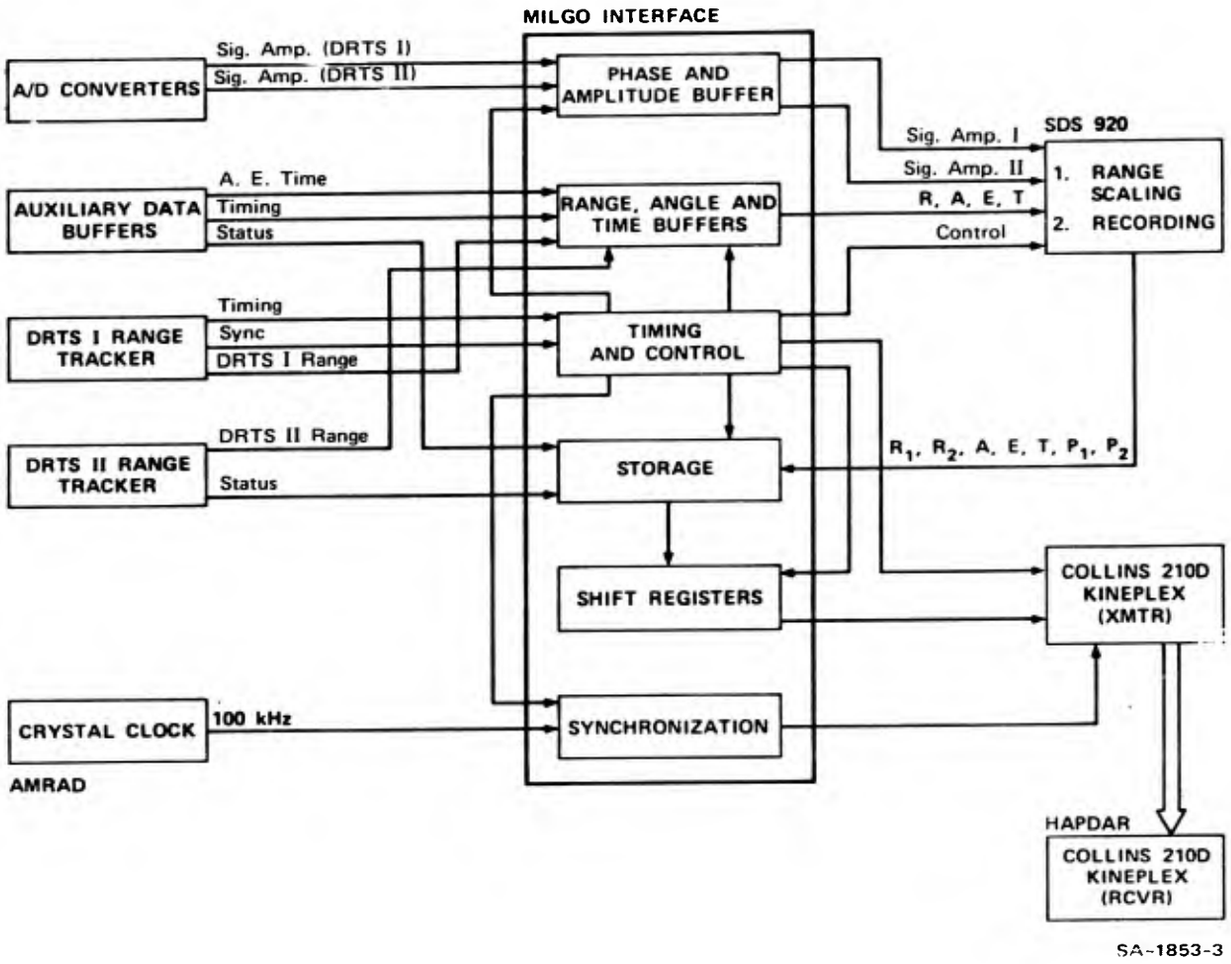


FIGURE 3 PHASE II CONFIGURATION (AMRAD/HAPDAR INTERFACE)

From the MILGO equipment the data are sent to the SDS 920 computer for scaling and formatting for transmission to HAPDAR. The program that accomplishes these operations is described in Appendix A.

After the calculations are completed in the SDS 920, the processed data are sent back to the MILGO equipment to be placed in storage registers for transmission to HAPDAR. From the storage registers the data is gated into shift registers to drive the kineplex modem, a Collins Model 210 D, for transmission by dedicated land lines to the kineplex modem at HAPDAR.

C. HAPDAR

The HAPDAR radar is a multifunction array radar designed, constructed, and operated by the Sperry Gyroscope Division of Sperry Rand Corporation. A good description of the radar is available in the literature.[3] Its characteristics are summarized in Table 2.

Table 2

HAPDAR RADAR CHARACTERISTICS

Frequency	1350 MHz
Peak power	2 MW
PRF	variable
Antenna gain	35.9 dB
Antenna beamwidth	1.6 degrees
Pulse length	20 μs chirped to 10 MHz
Beamswitch time	100 μs
S/N on 0.1 m^2 target at 100 km	15 dB
Range accuracy	2.3 meters
Angle accuracy	0.6 mrad

The HAPDAR radar is controlled by an IBM 360/65 computer that orders radar transmissions and specifies the receiver gates, thresholds, and attenuations for reception of the transmissions. The rate of transmission is a variable under control of the computer, but the upper limit (set by transmitter constraints) is about 300 pulses per second. The nominal rate for track was 20 pps, and whatever resources remained after servicing all track requests were allocated to search.

For the SRI experiment, two track channels were dedicated to tracking the two targets designated by AMRAD. The first track channel was dedicated to the AMRAD primary tracker. This first channel used a split gate tracker with a gate width of ±0.4 μsec and a false alarm probability of 10^{-2}. The initial receiver attenuation setting was determined from the predicted signal strength from the AMRAD data.

The second track channel was dedicated to the AMRAD secondary, range-only tracker. This channel used a split gate tracker with a gate width of ±2 μsec and a false alarm probability of 1.6×10^{-2}. The larger gate width was required because the uncertainty of measurement was quite large in angle for the range-only tracker. Here again the initial receiver attenuation was set based on the signal strength predicted from AMRAD measurements.

The relative level of the difference in sensitivity of AMRAD and HAPDAR was measured on 7 February 1972 by tracking a six-inch balloon-borne sphere with both radars. This measurement indicated that AMRAD was 24 dB more sensitive. Because of the possibility of locking-up on clutter, the predicted signal strength was scaled by -23 dB rather than -24 dB, leaving a 1-dB margin.

Besides the dedicated track channels, HAPDAR conducted a search about the two targets designated by AMRAD. In this way, it was hoped to

generate tracks for HAPDAR's track file that could then be correlated with the state vectors of the handed-over targets.

D. IBM 360/65

In early 1972, ABMDA installed an IBM 360/65 general purpose digital computer at HAPDAR as a part of the ABMDA Data Processing Field Test Facility. This third generation machine has a memory cycle time of 0.75 μsec and a million words of central processor memory. It uses eight bits per byte and four bytes to the word. Its execution speed for a fix point addition is 1.3 μsec.

RCA, as the operating contractor at the Data Processing Field Test Facility, modified the ABMDA supplied baseline software and installed this modified software to permit real-time control of the HAPDAR radar.

The baseline software known as post-PHSD was developed for ABMDA by IBM in order to demonstrate that commercially available hardware and software could be tailored to satisfy the requirements of ballistic missile defense data processing. IBM developed a real-time control program/operating system known as BMDOS and designed and implemented the software for tactical algorithm and process construction for BMD. This development is described in a nine volume report.[4]

For this contract, SRI developed or modified several algorithms for use in processing handover data from AMRAD. These algorithms, which are described in the next section, were integrated into the tactical process of the post-PHSD by RCA. A description of the post-PHSD SRI handover software interface is given in Appendix E. This software combination permitted the IBM 360/65 computer to control the HAPDAR radar in (1) search, target verification, track initiation, and track and (2) acquisition of target tracks based on AMRAD data.

E. SRI Software Development

1. Introduction

A block diagram of the AMRAD-to-HAPDAR handover, correlation, and search system is shown in Figure 4. The program AMTOHAP, which resides in AMRAD's SDS 920, receives interrupts from the AMRAD radar; the program responds to these interrupts by either inputting data measurements from the AMRAD radar or outputting them to the HAPDAR radar. (The program AMTOHAP is described in Appendix A.) The signals sent to HAPDAR travel via the kineplex link, the RDT, the RCI, and the parallel data adapter (PDA). An RCA channel-control program then reads the data into fast memory--that is, the AMRAD data file--from the PDA.

The SRI program TRANFT takes the data from the AMRAD data file, converts it to HAPDAR coordinates, and fits it with appropriate filtering algorithms. Following transformation and fitting, the program TRANFT outputs the filtered data to an AMRAD track file and a QUECOR data file. The AMRAD track file is used by the HAPDAR radar to initialize the two channels it has dedicated to handover messages. Similarly, the QUECOR data file is used to provide information for correlating AMRAD and HAPDAR tracks with QUECOR, as well as to generate search-sector requests by SSERCH for the HAPDAR radar. (All the input and output from TRANFT, QUECOR, and SSERCH occurs via FORTRAN subroutines.)

2. HAPDAR Software for SRI Experiment

The block diagram in Figure 5 indicates the software developed by SRI and RCA to accomplish the handover experiments. RCA wrote the programs to take the AMRAD data from the RCI to the AMRAD data file, RSDS, in the IBM 360/65 computer.

SRI developed the three programs TRANFT, QUECOR, and SSERCH. TRANFT smooths the AMRAD data, QUECOR correlates the AMRAD data with the

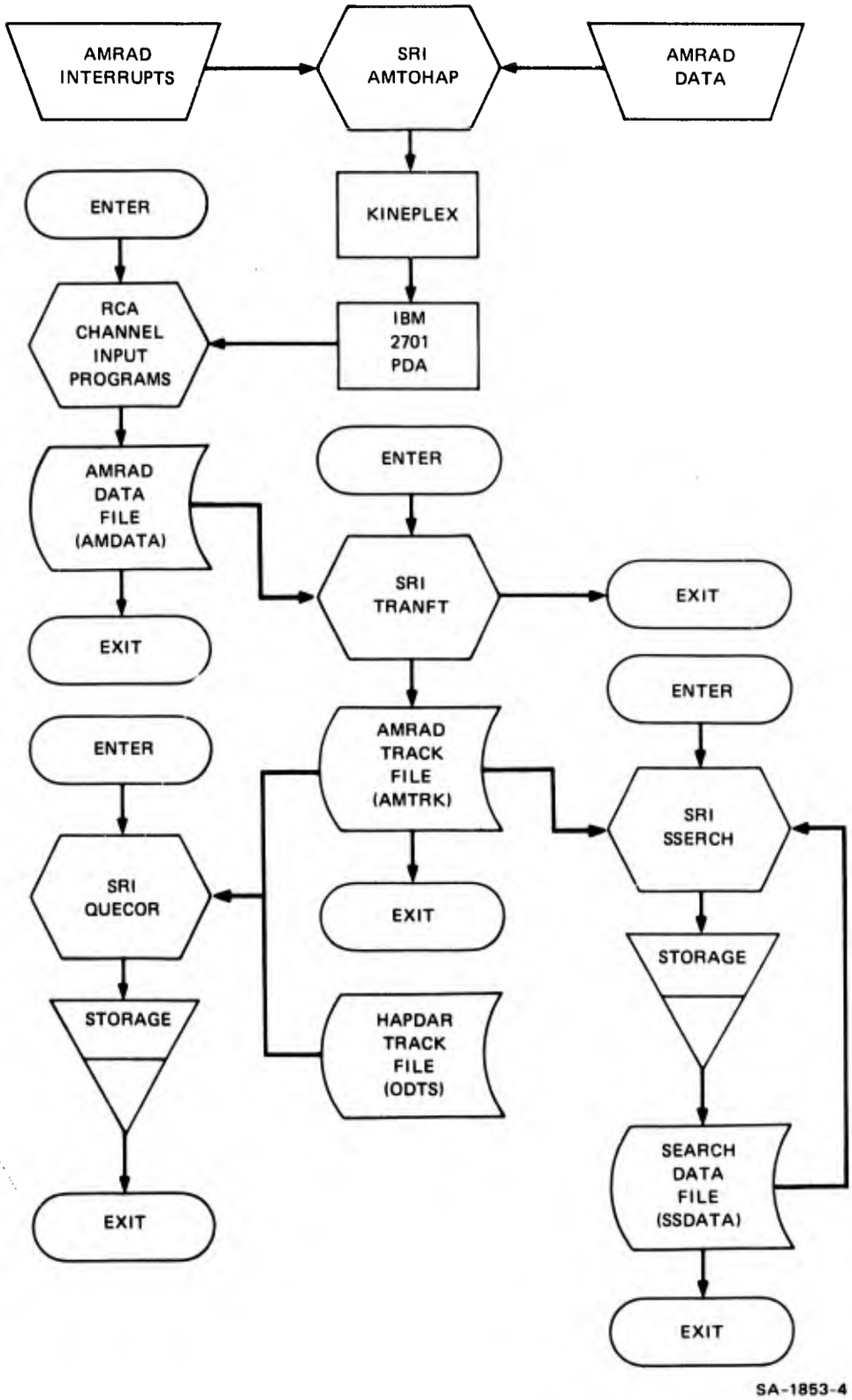


FIGURE 4 BLOCK DIAGRAM OF HANDOVER, CORRELATION, AND SEARCH SOFTWARE

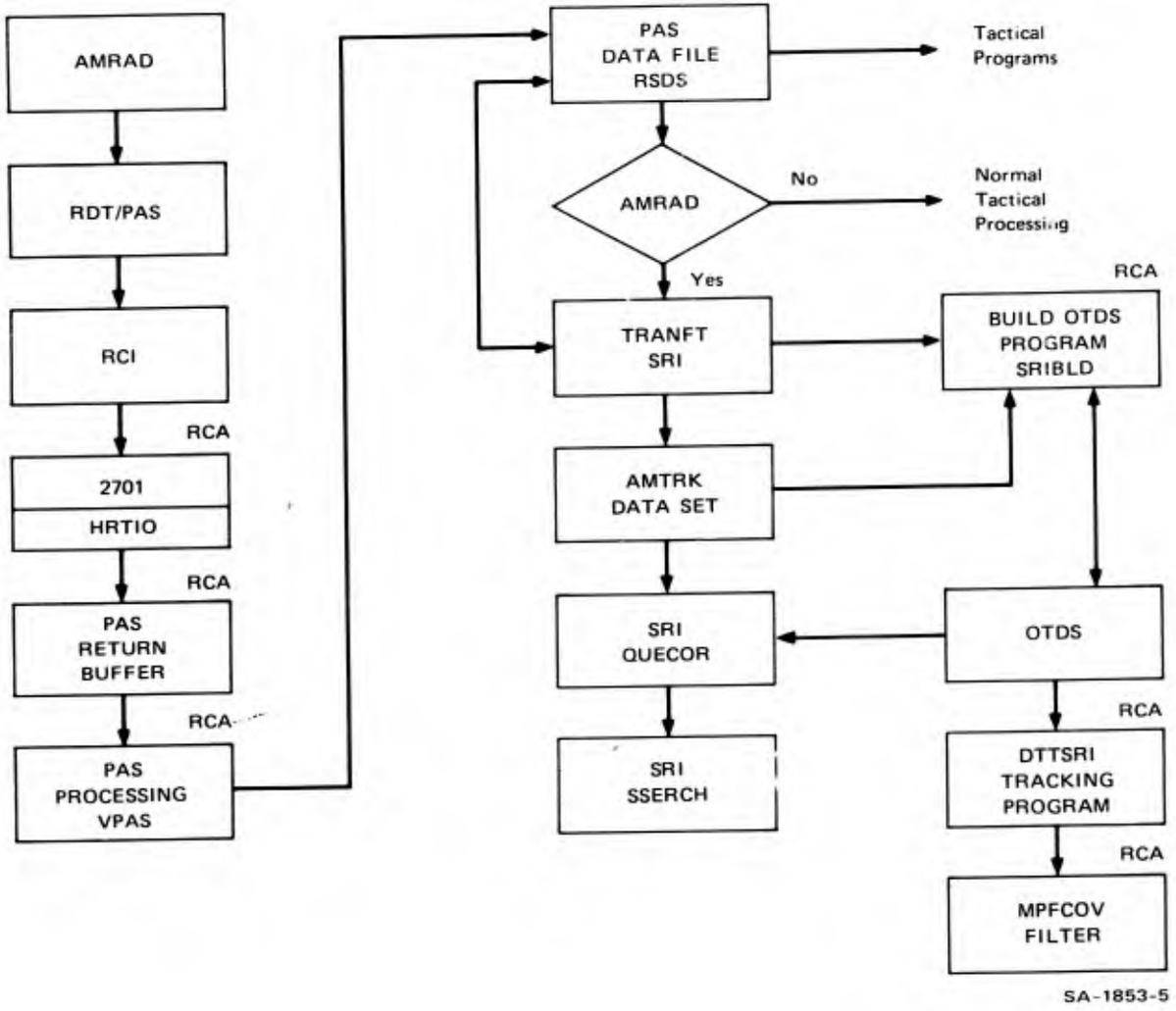


FIGURE 5 HAPDAR SOFTWARE FOR SRI HANDOVER EXPERIMENT

track file (OTDS), and SSERCH generates search beam positions for limited sector search handover.

To integrate these programs with the other software, RCA developed SRIBLD, DTTSRI and MPFCOV. SRIBLD took the current state vector from AMTRK, a data set constructed by TRANFT, and initially built an instance in the track file, OTDS, or reinitialized the track in OTDS. DTTSRI was a special α-β track routine designed to update the tracks in the dedicated track channels.* The covariance for both the dedicated tracks and the other tracks in the OTDS using α-β tracking was calculated by MPFCOV. This calculation was identical to the method used by TRANFT (see Appendix B).

The constants for the α-β range trackers were $\alpha = 0.8$ and $\beta = 0.15$, and for the α-β angle trackers were $\alpha = 0.25$, $\beta = 0.1$. A Type I filter was used for AGC with $\alpha = 0.5$.

3. Program TRANFT

The program TRANFT converts the data received from the AMRAD radar to HAPDAR coordinates and then filters the resulting coordinates for a vehicle state vector and error covariance matrix. The program begins its transformation and fitting operations by inputting data from the AMRAD data file. This data includes the measured position coordinates range, elevation angle, and aximuth angle (R, E, and A), the range time (t_{new}), the signal-to-noise ratio (SNR), the data source (PAS or AMRAD), the target number (primary or secondary), and the tracking mode (manual, acquisition, autotrack). Following inputting of the AMRAD data, TRANFT

---

*For a description of α-β trackers see Ref. 5.

makes an immediate validity check (is $t_{new} > t_{old}$) and returns, without processing the data, to the operating system if new data has not been received.

The program TRANFT next converts the AMRAD R, E, and A measurements to equivalent HAPDAR R, u, and v measurements, where u and v are direction cosines. (The conversion routine was developed under the previous contract work and has been described in Ref. 2.) The vehicle state vector, Z_e, is next updated from time t_{old} to t_{new} with the transition matrix F. The updated or "predicted" state vector, Z_p, is then corrected using the transformed AMRAD measurements Z_m and a new estimated state vector, Z_e, thereby formed.

The program TRANFT next updates the error covariance matrix, E_e using the same transition matrix F. Following the updating operation, the measured errors $Z_m - Z_p$ are calculated, and a measured error covariance matrix, E_m, defined. The measured error covariance matrix is then averaged with the predicted one, E_p, and a new estimated error covariance matrix obtained. Following the above operations, the program TRANFT outputs the estimated vehicle position, the estimated vehicle velocity, the estimated AGC level, the track ID, and the time-of-validity to the AMRAD track file. Similarly, the estimated state vector, estimated error covariance matrix, track ID, and time-of-validity are outputted to the QUECOR track file.

A complete description of the program is given in Appendix B.

4. Program QUECOR

QUECOR attempts to discover which tracks from the HAPDAR track file are correlated with the current track in the AMRAD track file. This is accomplished by computing a correlation parameter Q_{ij} for the i^{th} AMRAD track and j^{th} HAPDAR track. A threshold Q_T is compared to each

Q_{ij} for each i and for all j. A Q_{ij} greater than Q_T indicates a decorrelation. Q_{ij} less than Q_T indicates a possible correlation between [illegible] and HAPDAR track j. Thus, if for a given i, there exists [illegible] that Q_{ij} is less than Q_T, we say that AMRAD track i is cor[illegible]AR track j.

The correlation parameter Q_{ij} is a function of the vector difference between the estimated position vector of AMRAD track i and the predicted position vector (i.e., predicted to the corresponding time of the AMRAD track estimate) of HAPDAR track j. It is also a function of the covariance matrix for this position difference vector. Let C_{mn}^{ij} be the elements of the inverse of this covariance matrix, and Δx_m^{ij} be the m^{th} coordinate of the position difference vector. Then we define

$$Q_{ij} = \sum_{m,n=1}^{3} C_{mn}^{ij} \, \Delta x_m^{ij} \, \Delta x_n^{ij}$$

When QUECOR is called, the first task is to obtain data from the AMRAD track file that will contain only one track. This data will be the latest handed over information. The subset of this data required by QUECOR consists of the estimated position coordinates of the tracked object, the corresponding three-by-three covariance matrix, and the time for which the estimate is made. QUECOR next obtains the data from the first HAPDAR track file. This data consists of the estimated position and velocity, the corresponding six-by-six covariance matrix, and the time for the estimate.

The HAPDAR track position and position covariance matrix (i.e., a three-by-three submatrix) are updated to the AMRAD track time. The position difference vector and its magnitude are then computed, and the two position covariance matrices are summed. Next, an upper bound on

the largest eigenvalue of this summed matrix is obtained by analyzing the diagonal elements and the magnitude of the off-diagonal elements. The ratio of the magnitude of the difference vector and the bound on the largest eigenvalue of the covariance matrix forms a lower bound on Q. If this lower bound exceeds Q_T, then its value is stored in an array and a flag is set in the corresponding element of a corresponding array to denote that the pair of tracks are decorrelated, and that this was determined by a lower bound calculation. When this happens, QUECOR proceeds to the next HAPDAR track and repeats the process.

In case the lower bound on Q is below Q_T, the exact value of Q must be calculated. This is done by first inverting the covariance matrix and then computing Q by Eq. (1). Again Q is compared to Q_T, and if greater its value is stored in the Q array and a second type of flag is set in the flag array to denote the decorrelation and the fact that Q was computed exactly. In case Q is less than Q_T it is also stored in the Q array and a third type of flag is set in the flag array to denote the possible correlation. QUECOR then proceeds to the next HAPDAR track and repeats the process. Finally, when all HAPDAR tracks are exhausted, QUECOR examines the flag array to determine if only one correlation flag is set. If so, then the index of that flag representing the identification number of the HAPDAR track is stored to indicate the correlation. Control of the CPU is then returned to the calling control program. Note that no real-time operation of the radar systems, nor any other tactical function, requires the outputs of QUECOR for the real-time field test. Thus, the outputs of QUECOR are simply stored and recorded for subsequent analysis.

A complete description of QUECOR is given in Appendix C.

5. Program SSERCH

SSERCH generates a series of beam pointing directions and range gates required for HAPDAR to search the volume behind a nuclear fireball that temporarily blocks out part of the AMRAD search volume. When SSERCH is called, the fireball is simulated within SSERCH by specifying a fireball radius and range along the line-of-sight from AMRAD to the most recent handed over state vector position coordinates. This range is set at a fixed distance in front of the handed over point. The fireball radius is also set at a constant value. The search volume is taken to extend from the front edge of the fireball to a maximum range defined by adding a fixed distance behind the simulated fireball. This maximum range corresponds to the limit to a tactical search sector for AMRAD.

SSERCH computes the direction of the simulated fireball from HAPDAR and takes this as the first required search beam direction, provided it falls within the off-boresight limits for HAPDAR. If this condition is not met, a flag is set that indicates that part or all of the required search volume falls outside the HAPDAR limits. However, new beam directions continue to be generated in case some of the later beam directions are feasible. Whenever the flag described above is set, SSERCH bypasses the calculation of the corresponding range gates and the setting of the beam coordinates into storage.

Each new beam is generated recursively from the previous beam direction by requiring the new beam direction to intersect the line-of-sight from AMRAD to the simulated fireball. In addition, the distance along this line-of-sight (between the two intersection points of the new and old beam) is required to be such that there is an overlap between the 3-dB contours of the beams. This overlap is specified in the sine space of HAPDAR where circles of constant radius define the beam contours, by

requiring the distance between the beam centers to be less than the diameter of the beam 3-dB circle.

Given the new or next beam direction, a minimum and a maximum range is computed so that the region specified by the intersection of the beam and the shadow tube of the fireball is included between the range limits. Provided that the beam direction is within the off-boresight limits of HAPDAR, the values of sin α, sin β, the minimum range, and the maximum range are stored for later transfer to the special search sector data set.

Beam directions are generated until the range of the intersection of a generated beam and the line-of-sight from AMRAD to the fireball exceeds the previously determined maximum range. This beam direction then becomes the last beam generated. However, in the case where at least one beam is generated within the off-boresight limits of HAPDAR and some subsequent beam falls outside these limits, the beam generation process is terminated when the off-boresight limit is exceeded. As before, a flag is set to indicate that the beams generated do not form a complete set.

In any case when beam generation ceases, the beam pointing coordinates are transferred to the search sector file, and control of the CPU is transferred back to the calling function.

A complete description of SSERCH is given in Appendix D.

F. Demonstration of Handover, Correlation, and Special Search

A test plan for field demonstration of target handover, correlation, and special search on request was prepared.[6] The plan included a method for calibrating the HAPDAR radar.

1. Calibration

It was necessary to calibrate the handover system to determine the magnitude and source of any static range and angle biases. To accomplish this calibration, the plan was to drop four 12-inch calibration spheres from an aircraft at the WSMR. The drops were to be as follows: one of the spheres at ORV, about 50 km in front of HAPDAR; one at George site, where the AMRAD line-of-sight is perpendicular to the HAPDAR line-of-site; and one each at Bell and UPDOC, to the left and right of HAPDAR's boresight, to check azimuth sensitivity.

Because of delays, the sphere drop aircraft were not available for calibration. Data from two balloon-borne spheres launched about 20 km due north of HAPDAR, near the boresight azimuth, were used for calibration. The winds carried the spheres east as they climbed in altitude. Data from these spheres were used to calibrate the range bias, the most critical bias for target handover. There was not an adequate spread in angle to calibrate the angle channels.

2. Demonstration of Handover and Correlation

Two kinds of demonstration were planned to demonstrate target handover and correlation. The first was the demonstration of handover of ballistic targets of opportunity from AMRAD track to HAPDAR track without HAPDAR searching for the target, and demonstration of the SRI developed track correlation algorithm using the handover track and the HAPDAR track file.

In these tests, targets of opportunity (such as PERSHING, sounding rockets, and preferably ATHENA, an ICBM reentry simulation fired from Green River, Utah) would be tracked by AMRAD, with both the DRTS I and II trackers. If there was any other object separated from the primary target, the range only tracker, DRTS II, would be placed on that

object. This track data would then be sent to HAPDAR, which would search about the targets to generate targets for its track file.

For handover and correlation, two of HAPDAR's track channels were dedicated to initializing on the smoothed handover data. These two channels were maintained in track at 20 pps. Every two seconds, these track channels were reinitialized with data from the handover track file.

The plan called for the independent tracks in the nondedicated channels to be tracked at 20 pps also. The state vectors and covariances of each target were maintained in the track file at HAPDAR. At handover times, once every second, the correlation measure, Q, was calculated using the data from the HAPDAR track file and the handover target state and covariance.

The second test was designed to demonstrate the ability of the SRI correlation algorithm to distinguish a particular target designated by AMRAD within a group of closely spaced radar targets. For this test, four 12-inch calibration spheres were to be dropped from an aircraft flying at 40,000 feet. The spheres were to be dropped at a rate of one every three seconds to form a group of radar targets spaced about 500 meters apart. AMRAD would designate one of the spheres to HAPDAR. Previously, HAPDAR would have searched and acquired the four spheres and the aircraft. The sphere designated by AMRAD would then be correlated with the five objects in HAPDAR's track file to identify the object designated by AMRAD.

Because the sphere drop aircraft was not available, the demonstration of target correlation with closely spaced spheres was not accomplished. The only ballistic target of opportunity that HAPDAR was able to track using the IBM 360/65 computer during the contract period was a sounding rocket, Rome-03. This mission is described in the next chapter and the results are analyzed in Chapter V.

3. Demonstration of Special Search on Request

The objective of the demonstration of Special Search on Request was to demonstrate the ability of a radar to search an arbitrary search sector on request of another remotely located radar, as might occur if the remote radar were "blacked out" by a fireball obscuring a region in its search sector. To accomplish this demonstration, 12-inch spheres were to be dropped in selected locations. AMRAD would track each sphere to determine the direction of the simulated obscured region. This direction would be communicated to the HAPDAR IBM 360/65 computer via the data link. At the 360/65 the special search algorithm would generate a search position data set along the line segment directed from AMRAD to the target, 20-km in length and centered on the target. This line would be searched once every second with the search continuing for ten seconds. After ten seconds the search would be reinitialized from a new designation message from AMRAD. The search hits would be recorded during the special search to demonstrate that the search was conducted properly.

Again this demonstration could not be completed because the sphere drop aircraft were not available and also there was not enough time available to complete checkout of the special search-on-request algorithm.

IV ROME-03 TEST DESCRIPTION

On 28 February 1973, handover and correlation on a moderate velocity target was accomplished. On this day, at 13:27:00 GMT, a sounding rocket (SRW-04/Rome-03) was fired from WSMR Launch Complex 33. AMRAD tracked this target throughout most of its trajectory with both the trackers using centroid tracking. AMRAD data during this flight were used to hand targets over to the HAPDAR radar and to correlate the handover data with the HAPDAR track file. In addition, AMRAD tracked a premission, balloon-borne 12-inch diameter sphere and also handed this target over to HAPDAR.

To describe this mission, pertinent data have been extracted from the AMRAD data summary.[7] The Rome-03 payload was boosted by a NIKE-HYDAC rocket to an altitude of 165.8 km where a small aerosol cannister was ejected, subsequently releasing a disc-shaped cloud of carbon particles. After cannister release, the payload was separated from the booster, followed by parachute deployment to slow the descent of the payload for recovery.

Figure 6 shows the ground track as recorded by AMRAD, with the times of significant events. Figure 7 shows the flight profile. Figure 8 shows the velocity history of the target and Figure 9 shows the target coordinates in range, azimuth, and elevation relative to AMRAD.

The radar cross section has been calculated from the AMRAD radar precursor pulse* returns, averaged over 0.1 sec, or five pulses. Figure 10

---

*The precursor pulse was 6 MW and 1.2 μsec.

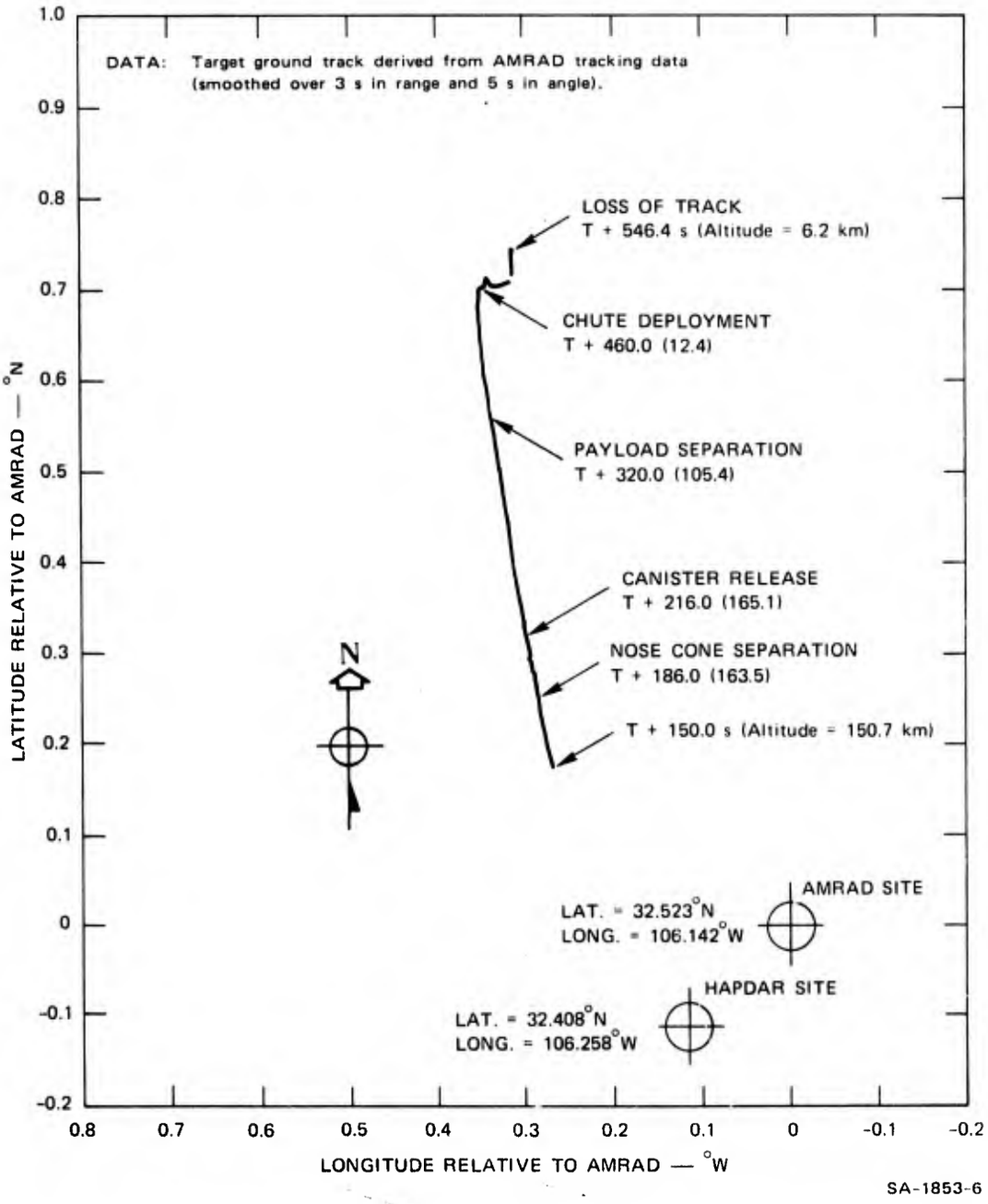


FIGURE 6 ROME-03 GROUND TRACK

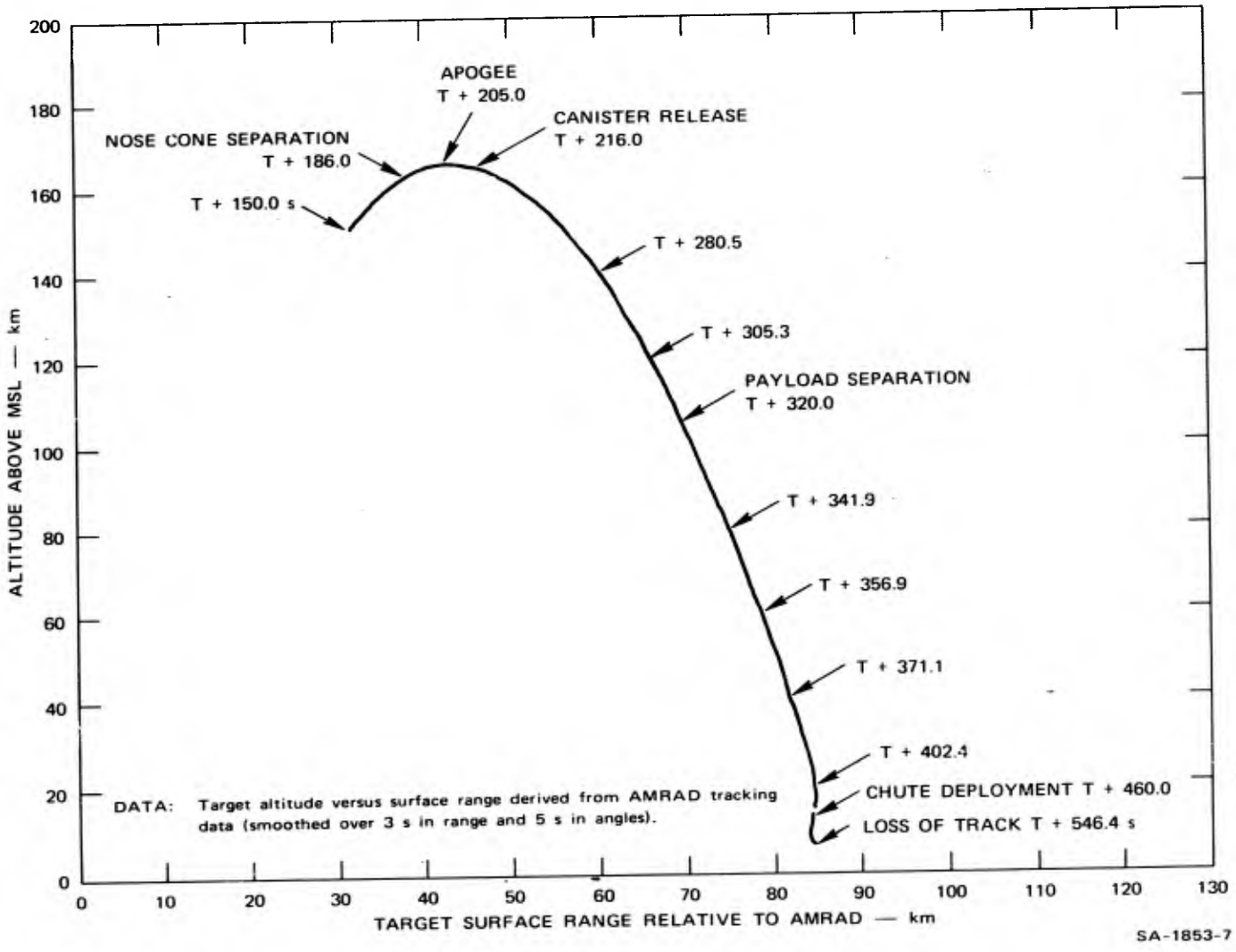


FIGURE 7 ROME-03 FLIGHT PROFILE

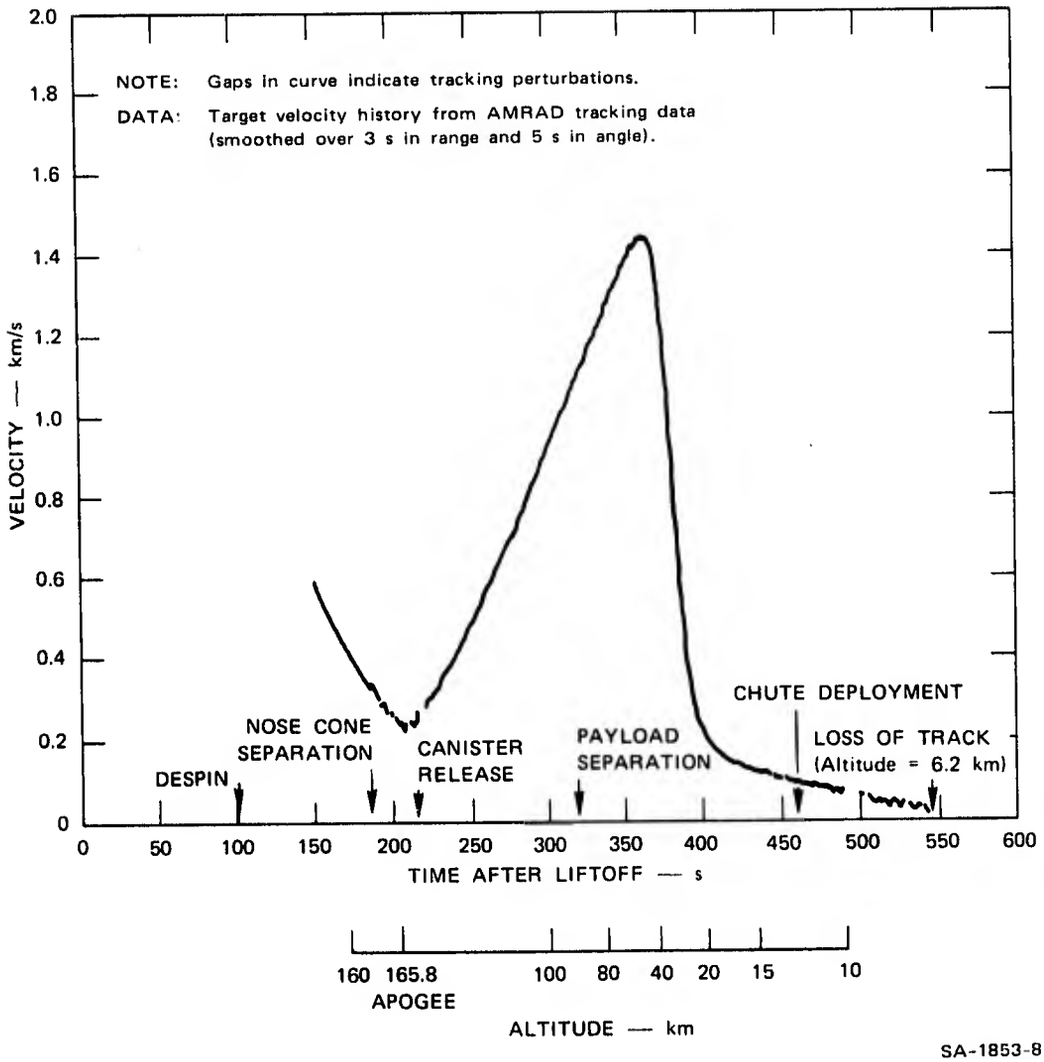


FIGURE 8 ROME-03 VELOCITY HISTORY

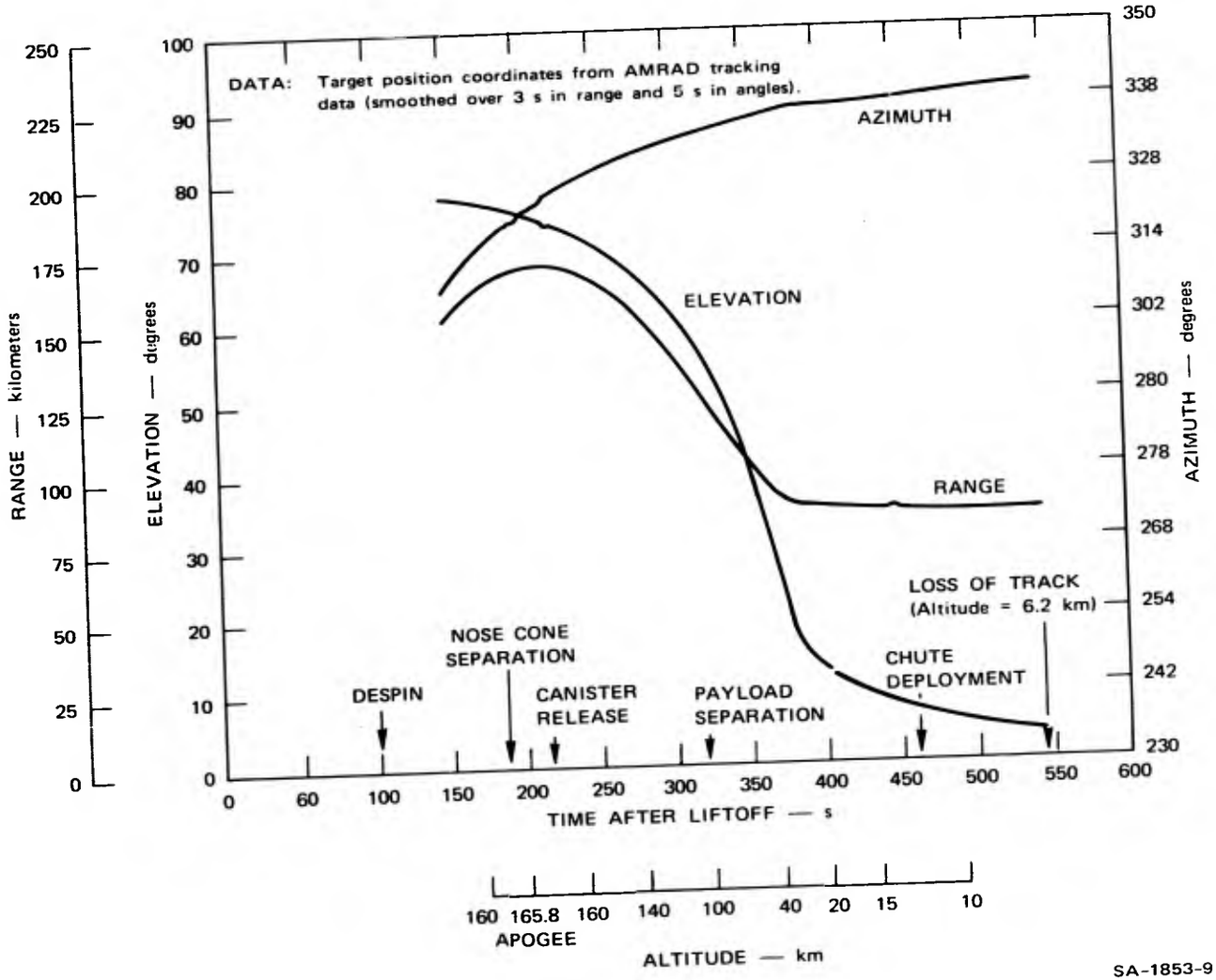


FIGURE 9 ROME-03 COORDINATES RELATIVE TO AMRAD

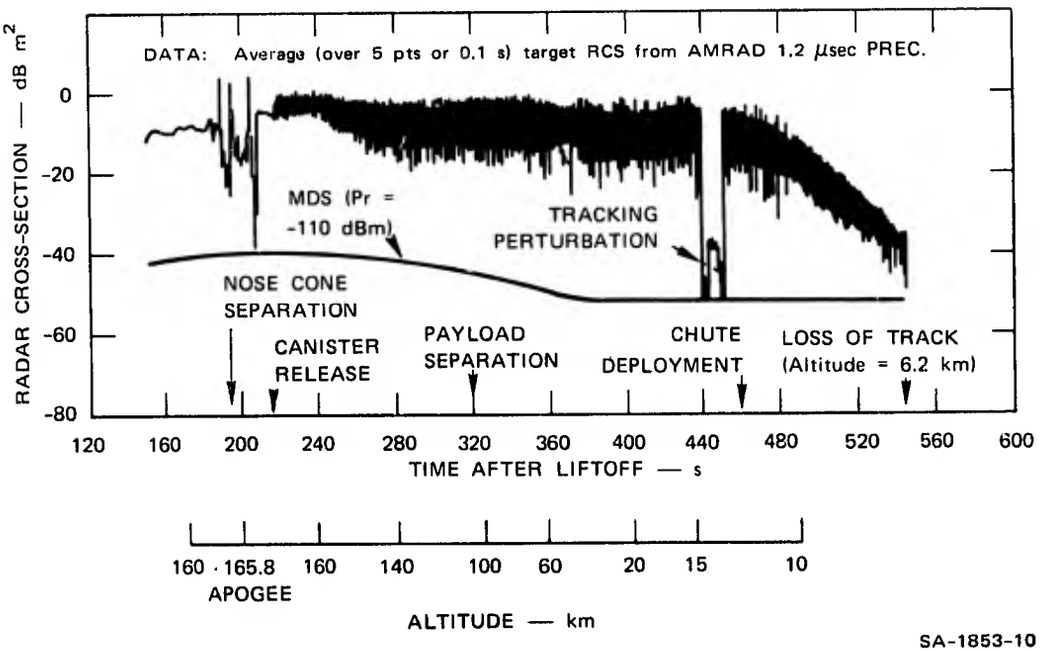
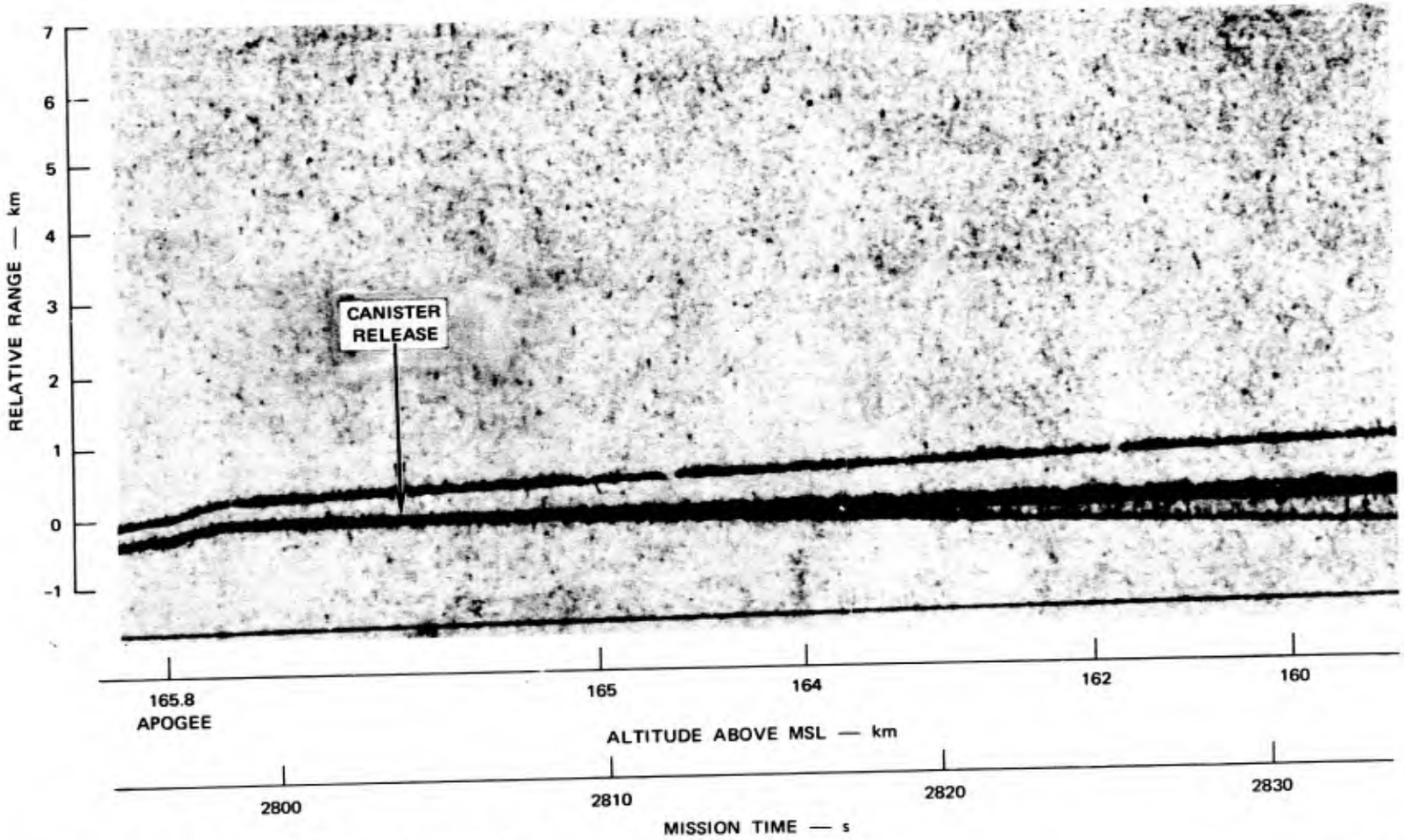


FIGURE 10 ROME-03 AVERAGE TARGET RCS

shows a plot of the radar cross section as a function of the time after lift off. The premission sphere calibration shows the calibration constant used in these computations to be within 1 dB of the measured value. Also shown in the figure is the equivalent cross section of the minimum discernible signal (MDS) at the range of the target. Considering that HAPDAR has about 24 dB less sensitivity than AMRAD, it can be seen that the signal-to-noise ratios at HAPDAR were quite low for Rome-03.

The I P pulse transmitted by AMRAD consisted of a six megawatt 0.1 μsec pulse repeated 50 times per second. The returns from this pulse were used to produce the range-time-intensity (RTI) record shown in Figure 11. Here only a selected portion is shown; the time is labeled



NOTE: Time of liftoff = 2590.7 seconds, mission time.

SA-1853-11

FIGURE 11 AMRAD-IP RTI RECORD

in HAPDAR mission time, with lift-off corresponding to 2590.7 seconds. HAPDAR collected data during the period 2821.5 to 2829.5 seconds. It can be seen that there are three targets visible during this period. The target in track is the Rome-03 rocket; the target farther in range is probably the nose cone. The target nearer in range diverging from the rocket is probably the canister. Thus during the period of the demonstration there were three targets available; however, all three were so low in radar cross section that HAPDAR could not independently acquire any of them.

V DATA ANALYSIS

A. Description of Data

Four sets of useful tracking data were obtained during the contract period. The first set of data was a track of a balloon launched sphere which took place on 2 February 1973. The total track duration extended over a period of about 262 seconds; data was recorded from five segments of this period, covering a duration of about 116 seconds. For convenience we will refer to this particular sphere as Sphere One. The ground track for Sphere One is shown in Figure 12. The average range and elevation relative to HAPDAR was 18,000 meters and 24°. The azimuth varied from 13.8° to 31° relative to HAPDAR. The elevation relative to AMRAD was greater than 24° since the ground range relative to AMRAD was much less than to HAPDAR.

The remaining sets of data were taken on 28 February 1973, and were associated with the Rome-03 mission described in Chapter IV. The first set of data was from a premission balloon-launched sphere, the second from the Rome-03 rocket, and the third from the parachuted Rome-03 instrumentation package toward the end of the mission. We will refer to these objects and their track data as the Rome Sphere, the Rome Rocket, and the Rome Payload, respectively.

The track on the Rome Sphere started at 654 seconds into the mission time, covered a period of 289 seconds, and data was recorded from three segments of this period covering a duration of 73 seconds. The ground track for the Rome Sphere is shown in Figure 13. The average range and elevation relative to HAPDAR was 57,300 meters and 12°. The azimuth

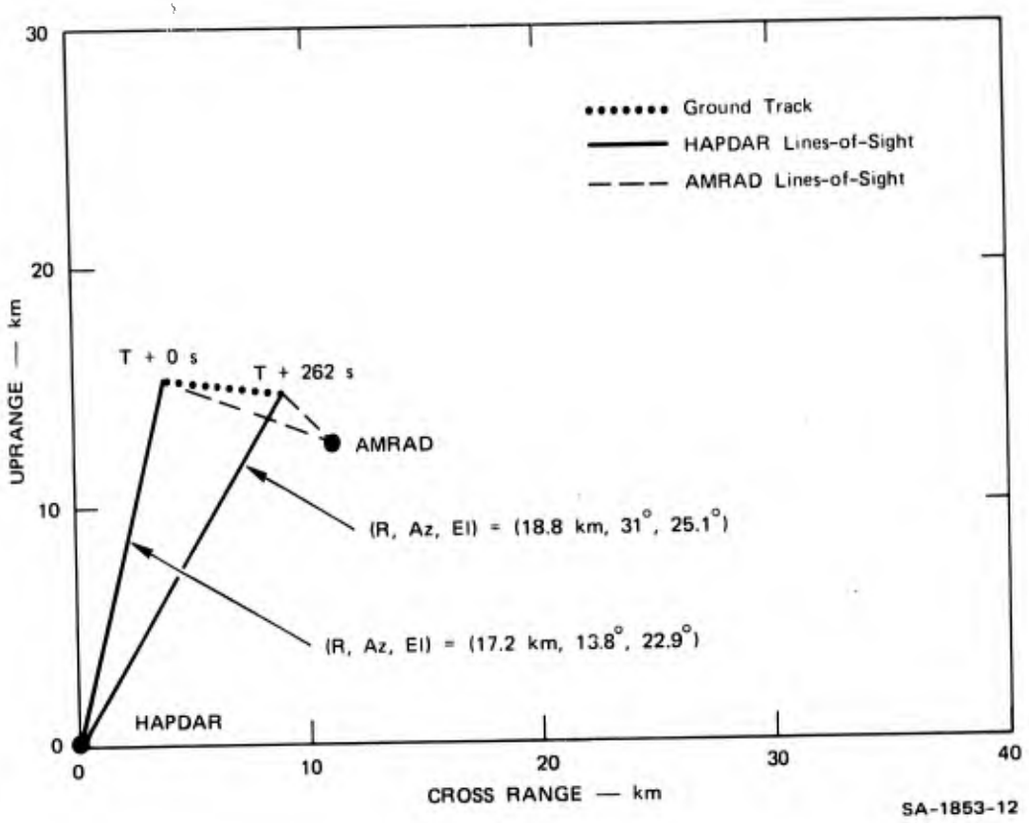


FIGURE 12 SPHERE ONE GROUND TRACK

varied from 4.5° to 15.8° relative to HAPDAR. The elevation relative to AMRAD was greater than 12° since the ground ranges relative to AMRAD were less than to HAPDAR.

The track on the Rome Rocket covered a period of eight seconds, starting from 2821.5 seconds into the mission time. The data consisted of only one segment. The ground track for the Rome Rocket is shown in Figure 14. The average range, elevation, and azimuth relative to HAPDAR were 170,000 meters, 70°, and -18°, respectively. Since the ground range relative to AMRAD was slightly less than the ground range from HAPDAR, the elevation relative to AMRAD was slightly greater than 70°.

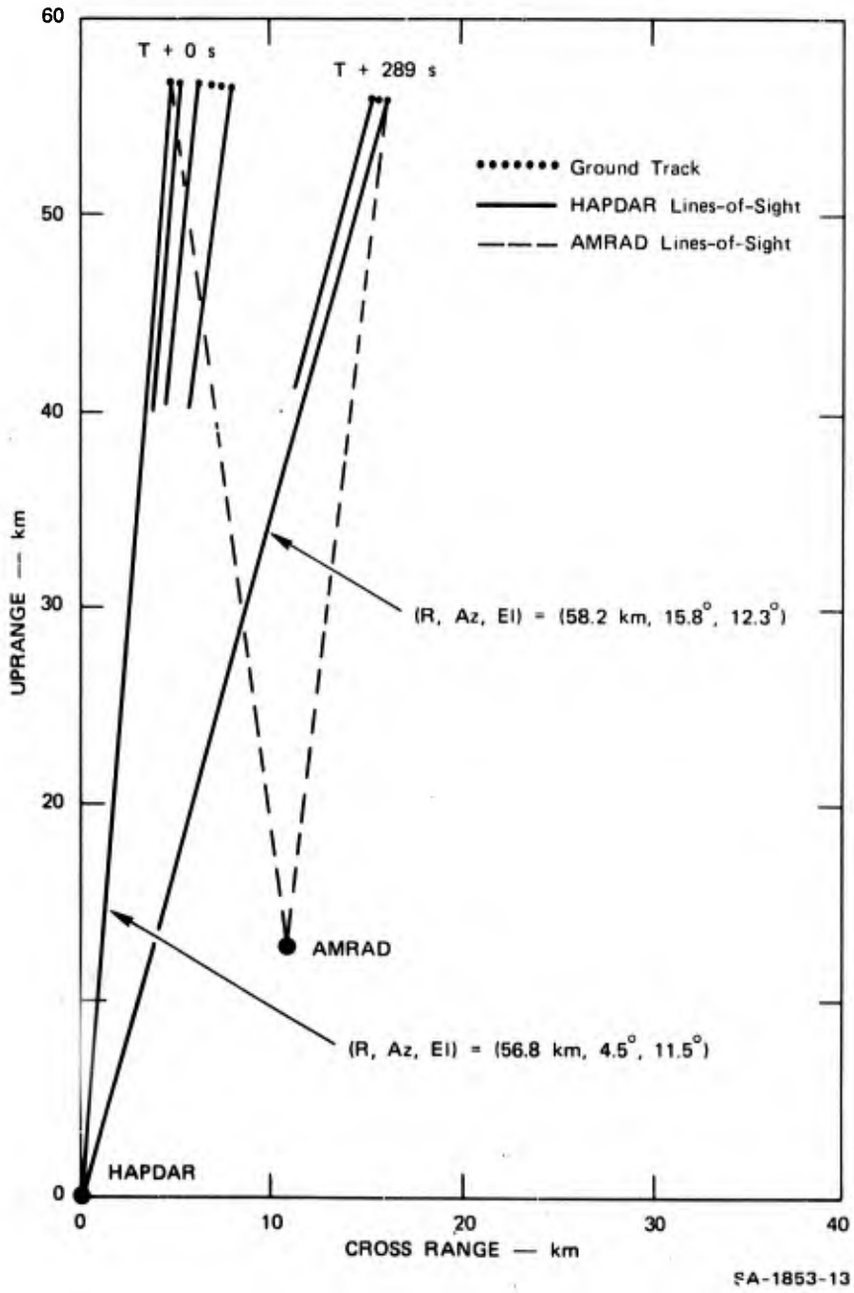


FIGURE 13 ROME SPHERE GROUND TRACK

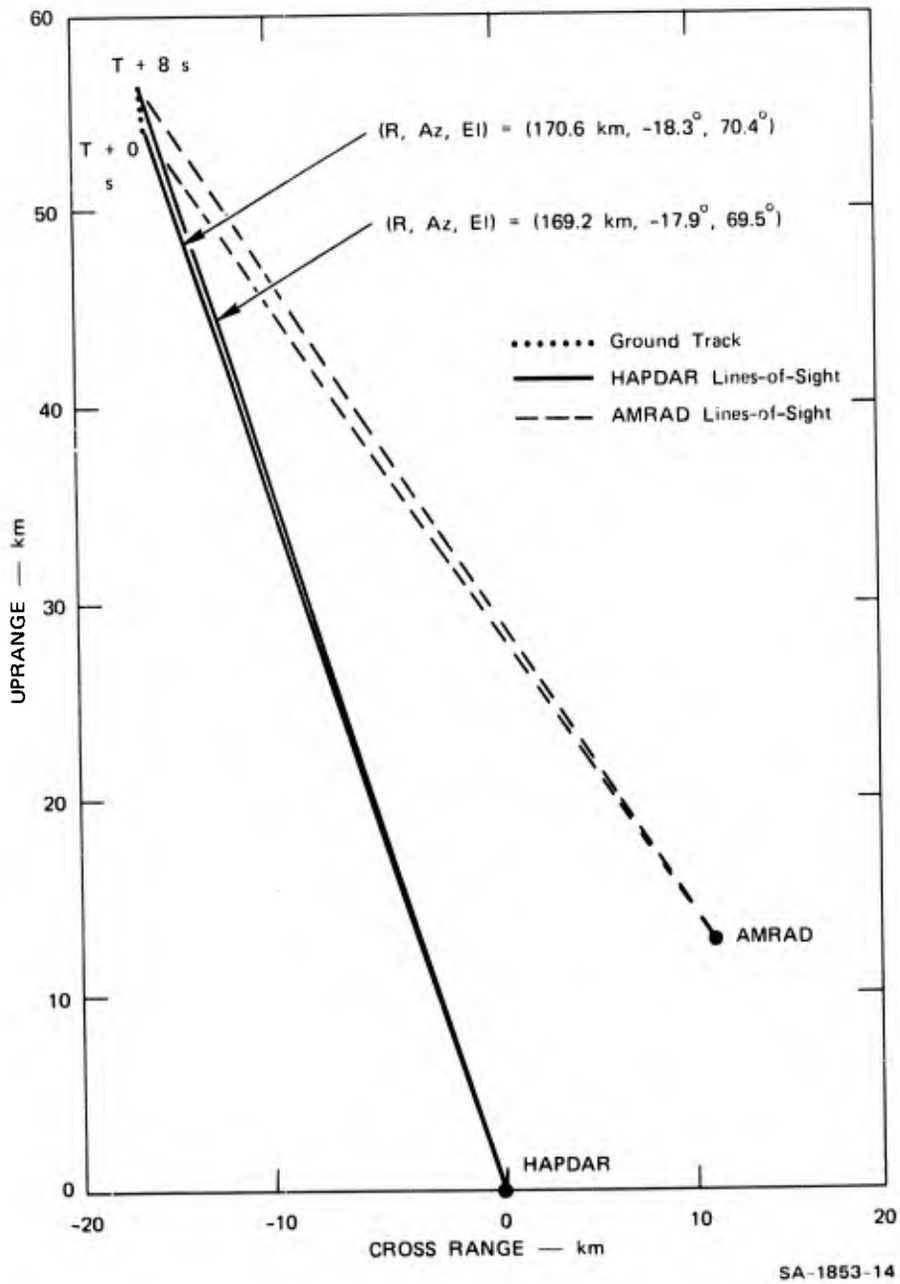


FIGURE 14 ROME ROCKET GROUND TRACK

The track on the Rome Payload covered a period of 55 seconds and consisted of only one segment of data. Since the PHSD software was abnormally terminated while the Rome Rocket was being tracked, the system was reloaded and restarted with the result that the mission time was also restarted. The new mission time for the start of the Rome Payload track data was approximately 43 seconds. The ground track for the Rome Payload is shown in Figure 15. The average range, elevation, and azimuth from HAPDAR were 93,800 meters, 6°, and -12.5°, respectively. The elevation relative to AMRAD was about 6.5°, only slightly greater than the elevation relative to HAPDAR. At this elevation angle the AMRAD beam is beginning to partially pass through the clutter fence; it is estimated that, for this particular geometry, an attenuation of about 0.75 dB can be expected.

B. Handover Calibration Results

The test plan for obtaining relative calibration data between AMRAD and HAPDAR called for four controlled sphere drops from an aircraft. The altitudes and locations of these sphere drops were designed so that the measurements would cover a good spread in azimuth, elevation, and range. The resulting data were to be used in fitting a linear model of relative R, u, and v bias between the two radars. It was learned during the week of 8 January 1973 that the aircraft scheduled to provide these sphere drops was no longer available, with the result that the calibration test plan could not be carried out. Instead, it was hoped that calibration data would be obtained on various balloon-launched spheres and other targets of opportunity.

Some data useful for first order calibration were obtained from the track data of Sphere One, Rome Sphere, Rome Rocket, and Rome Payload. The data obtained on Sphere One indicated an average range difference between AMRAD and HAPDAR of 23 meters. The range measured by AMRAD (transformed to HAPDAR coordinates) was consistently greater than the HAPDAR

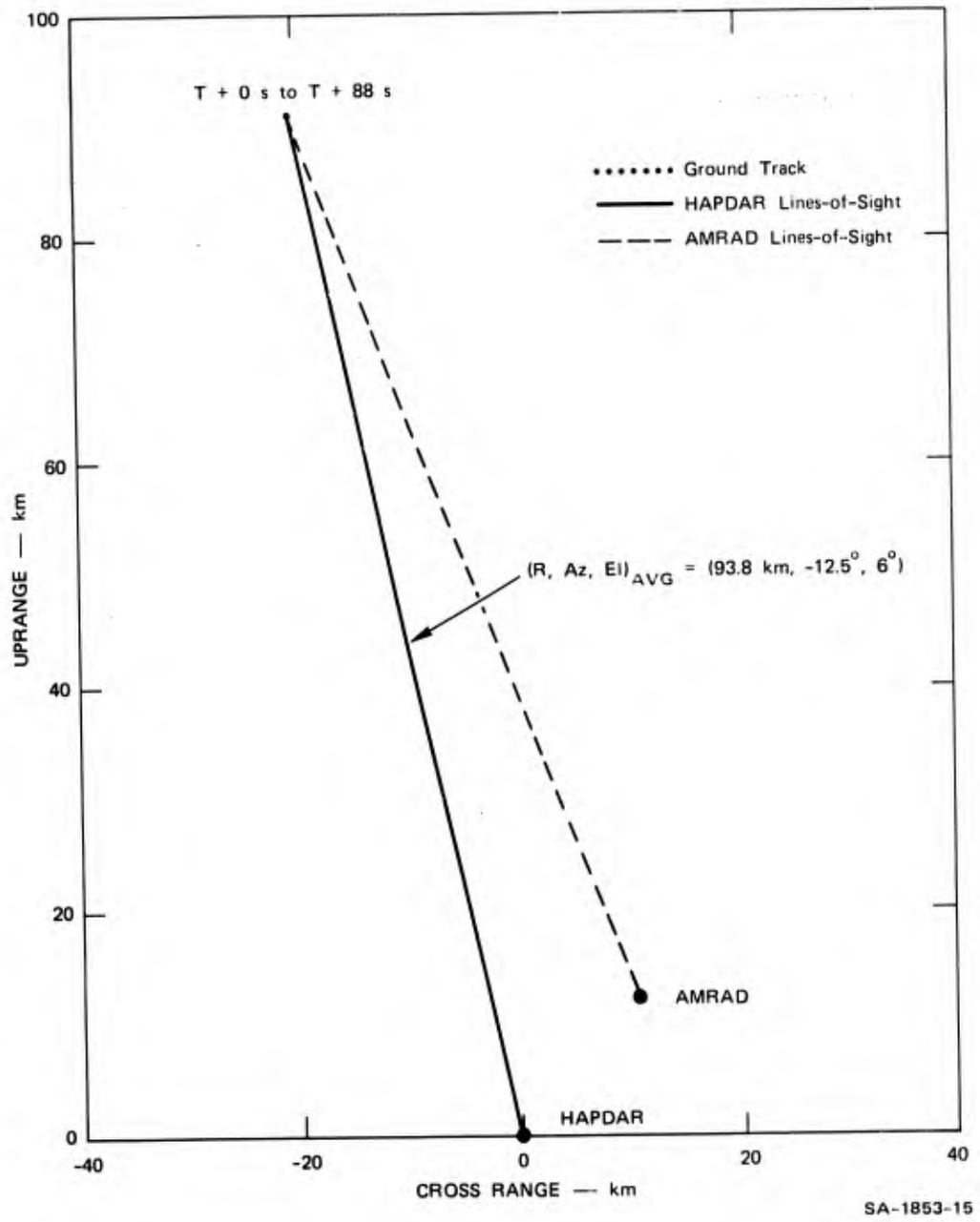


FIGURE 15 ROME PAYLOAD GROUND TRACK

measurements. In addition, there was an average u difference of 2.6 millisines and an average v difference of -1.8 millisines.

Comparing the biases in R, u, and v described above to the range gate and beamwidth of HAPDAR indicates that the range bias is about 38 percent of half the range gate width, the u bias is about 18 percent of half the beamwidth, and the v bias is about 12 percent of half the beamwidth.

Since the data indicated that the range bias was the most crucial to the handover experiments, it was decided to correct all HAPDAR measurements for all operations subsequent to 2 February 1973.

The angle bias terms were not included in the HAPDAR measurement corrections at this time for several reasons. First, they were not very crucial to the success of the handover experiment. Second, they were obtained at an azimuth angle far removed from the Athena mission azimuth (about -20°) and previous experience indicated that the angle biases appeared to be functions of the angles themselves.

The remaining data useful for calibration consist of the data on the Rome mission tracks. Due to the fact that these data were the last data to be obtained on the radar netting program, most of the reason for the handover calibration had vanished. However, since the SRI handover software may be used at HAPDAR for some additional experiments, the data were analyzed for range bias.

The range biases for the Rome Sphere, Rome Rocket, and Rome Payload were obtained by averaging the difference between the AMRAD first tracker range estimate and the corresponding HAPDAR track range estimate taken at one second intervals from the available data. For the Rome Rocket, nine smoothed points were used which covered the total period of track data available. Twenty points and 30 points were used from the data on

the Rome Payload and Rome Sphere, respectively. Although these numbers of points do not cover the entire track periods for these two tracks, the averages were quite stable after about ten points.

The result of the bias analysis was that a bias of 0.4 meters was obtained from the Rome Sphere data, 19.2 meters from the Rome Payload data, and 54.5 meters from the Rome Rocket data. These three sets of data were taken at average ranges of 58,000 meters, 94,000 meters, and 170,000 meters, respectively. These biases were such that AMRAD measurements were greater than HAPDAR measurements for each case. A linear fit to these biases and the bias numbers themselves are shown in Figure 16. The linear fit gives a bias of -28.5 meters at zero range and a slope of 0.000491 meters/meter. Thus, the bias is zero at a range of about 58,000 meters. As can be seen from Figure 16, the three points fit a linear model very well. It should be noted that these biases were obtained after the 23-meter bias compensation had been applied to all HAPDAR measurements. Thus, any range bias compensation based on the results shown in Figure 16 would be in addition to the previous 23-meter compensation.

If we were to adjust the results to include the 23-meter compensation, we would have to add 23 meters to all points, and we would obtain the linear bias model shown in Figure 17. Note that this linear model has a small zero range axis intercept. However, if we add the Sphere One bias data, as we have done in Figure 17, we get a point that falls significantly above the linear model and suggests a piecewise linear model. Thus, the Sphere One bias data and the Rome data appear to be somewhat inconsistent. It is not clear at this time what phenomenon might cause this type of result.

The linear trend suggests at first hand an incorrect velocity of light, or a slow or fast clock in the system. However, these have been

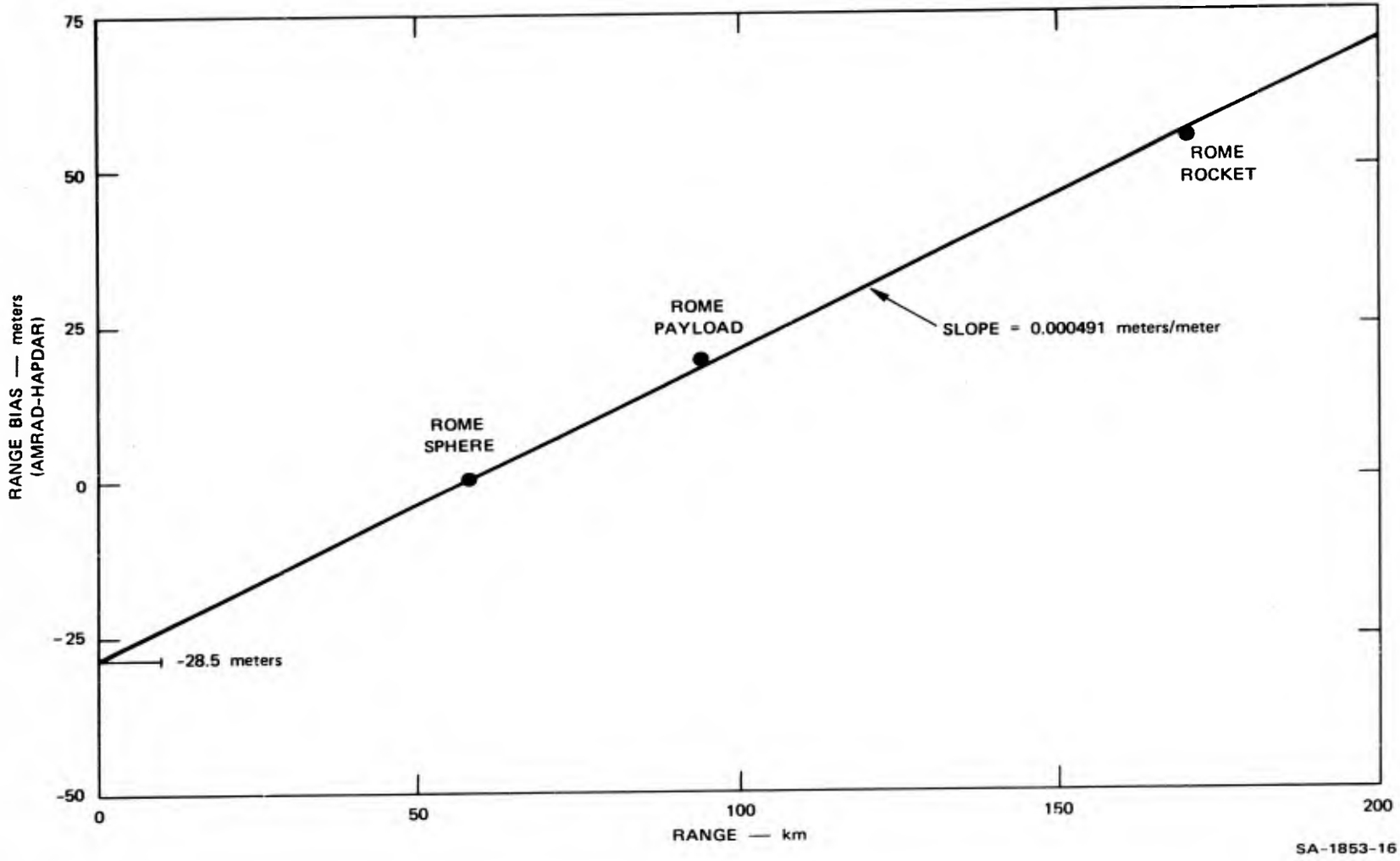


FIGURE 16 LINEAR RANGE BIAS MODEL BASED ON ROME-03 DATA

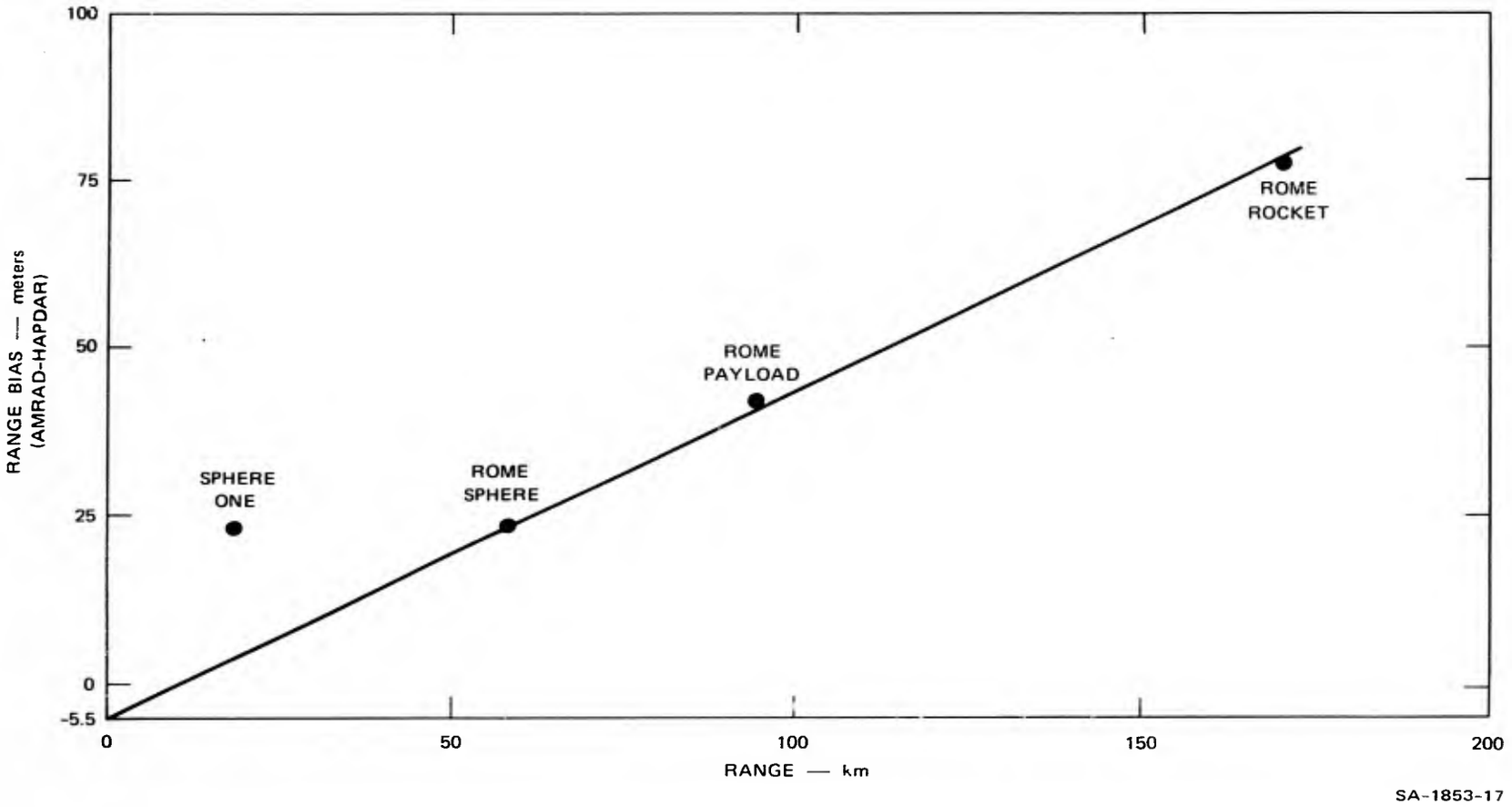


FIGURE 17 ADJUSTED RANGE BIAS

checked and rechecked and appear not to be the problem. It is also possible that the range bias might be affected by the direction of the beam, since the elevation of the Rome Rocket data was different from the elevation of the other data points. However, it is difficult to postulate such a mechanism. Finally, it should be noted that the Sphere One data and the Rome data were obtained approximately 20 days apart, a time during which a number of radar hardware changes and various software modifications were made to the system.

The bias results discussed above were passed on to the RCA people for their information, and TRANFT--which processes handed-over AMRAD data--was adjusted so that the linear range bias corrections shown in Figure 16 will be applied to any operation of the handover software at HAPDAR subsequent to 16 March 1973.

C. Handover Results

The data from the Rome-03 mission was analyzed to assess the degree of success of the handover experiment. One of the important factors in the handover experiment is the handover delay time. By handover delay time we mean the time between the time of validity of the handed over AMRAD state and the time when the first handover pulse transmitted by HAPDAR reaches the handed over object. For the Rome-03 mission, handover delay times of from 70 to 130 milliseconds were obtained. Most often, the delay was only 70 ms. The variation in delay is a function of the PHSD scheduling load at any particular time. About 50 ms of the delay time is used in passing the AMRAD data to HAPDAR. Thus, we see that generally only 20 ms was required to transmit the first handover pulse. This indicates that handover delays were within acceptable limits.

The data obtained on the handover of the Rome Sphere are tabulated in Tables 3 and 4. Table 3 contains the handover data for the first

Table 3

AMRAD FIRST TRACKER HANDOVER DATA FOR ROME SPHERE

Attempt Number	Handover Time (seconds)	Predicted SNR (dB)	Time of 1st Hit (seconds)	Measured SNR (dB)	Range Error on 1st Hit (meters)	Number of Misses Before 1st Hit
1	654.96	28	655.01	25.3	1.9	0
2	656.96	28.9	657.03	28.3	6.6	0
3	658.96	28	659.03	26.3	3.8	0
4	660.96	28	661.03	26.6	-2.1	0
5	662.96	28	663.03	27.1	1.7	0
6	682.96	28	683.03	24.7	-0.9	0
7	684.96	28.5	685.03	26.0	-2.1	0
8	686.96	28	687.03	27.1	5.7	0
9	688.96	29	689.03	25.7	3.8	0
10	690.96	28	691.03	26.0	2.6	0
11	692.96	28.5	693.03	27.4	-1.9	0
12	694.96	28	695.01	25.8	3.5	0
13	696.96	28	697.03	26.9	-6.1	0
14	698.96	28	699.03	27.7	5.9	0
15	700.96	28	701.07	24.9	1.4	1
16	702.96	28.8	703.03	25.5	-5.7	0
17	704.96	28.9	705.03	27.5	3.3	0
18	706.96	28	707.03	26.3	1.9	0
19	708.96	28	709.09	24.7	5.2	1
20	710.96	28	711.03	25.8	-2.6	0
21	712.96	28.2	713.03	26.2	0.003	0
22	714.96	28	715.03	26.3	-2.8	0
23	912.13	28	912.3	25.5	8.5	1
24	914.13	28.9	914.3	24	-5.4	1
25	916.13	28	916.22	24.9	0.24	0
26	918.13	28.5	918.2	25.4	4.3	0
27	920.13	28	920.3	23.9	-6.9	1
28	922.13	29	922.3	24	3.3	1
29	924.13	29	924.24	26.5	-2.1	0
30	926.13	28	926.24	25.2	-0.95	0
31	928.13	28	928.24	24.7	-3.1	0
32	930.13	28	930.3	23.6	-4.3	1
33	932.13	28.9	932.24	26.4	7.6	0
34	934.13	30	934.32	24.8	2.8	2
35	936.13	28.2	936.24	26.3	6.4	0
36	938.13	28.1	938.24	25.6	-0.24	0
37	940.13	28	940.2	25.3	3.3	0
38	942.13	28	942.24	25.5	2.6	0

Table 4

AMRAD SECOND TRACKER HANDOVER DATA FOR ROME SPHERE

Attempt Number	Handover Time (seconds)	Predicted SNR (dB)	Time of 1st Hit (seconds)	Measured SNR (dB)	Range Error on 1st Hit (meters)	Number of Misses Before 1st Hit
1	654.10	29.5	654.15	26.7	20.8	0
2	656.10	30	656.27	24.5	30.2	1
3	658.10	29	658.21	26.8	30.2	0
4	660.10	29.9	660.21	24.7	30.2	0
5	662.10	29.5	662.17	25.1	30.2	0
6	664.10	30	664.21	27.5	30.2	0
7	684.09	30	684.21	25.8	30.2	0
8	686.09	29.9	686.17	26.1	30.2	0
9	688.09	29.9	688.21	24.7	30.2	0
10	690.09	29.8	690.17	26.2	30.2	0
11	692.10	29.7	692.17	24.5	30.2	0
12	694.10	29.8	694.23	23.5	30.2	1
13	696.10	29	696.23	26.2	30.2	0
14	698.10	29.3	698.19	27.6	30.2	0
15	700.10	29	700.21	27.1	30.2	0
16	702.10	29.5	702.21	26.5	30.2	0
17	704.10	30	704.27	22.9	30.2	1
18	706.10	30	706.21	27.2	30.2	0
19	708.10	29.9	708.21	25.8	30.2	0
20	710.10	29.9	710.27	24.6	30.2	1
21	712.09	29.8	712.21	27.2	30.2	0
22	714.09	29.5	714.21	28.3	30.2	0
23	913.07	30.1	913.20	23.5	30.2	1
24	915.07	29.3	915.14	26.2	30.2	0
25	917.07	30	917.20	24.8	30.2	1
26	919.07	30	919.20	25.9	30.2	1
27	921.07	30.2	921.20	23.6	30.2	1
28	923.07	30	923.20	24.7	30.2	1
29	925.07	30	925.20	24.8	30.2	1
30	927.07	30	927.20	24.2	30.2	1
31	929.07	30	929.20	23.8	30.2	1
32	931.07	30	931.22	22.6	30.2	1
33	933.07	30	933.24	24.5	30.2	1
34	935.07	29.8	935.16	25.7	30.2	0
35	937.07	30.2	937.16	25.2	30.2	0
36	939.07	30	939.24	25.6	30.2	1
37	941.07	29.9	941.22	25.5	30.2	1
38	943.07	29.9	943.16	24.7	30.2	0

AMRAD tracker, and Table 4 for the second AMRAD tracker. The following information is given in these tables:

(1) Count of handover attempts

(2) Time of handover request data

(3) Predicted SNR for HAPDAR

(4) Time of first hit by HAPDAR

(5) Measured SNR for first hit

(6) Measured range error for first hit

(7) Number of misses before first hit.

Table 3 shows that 38 handover attempts were made for AMRAD's first tracker. Of these 38 attempts, 30 hits were obtained by HAPDAR on the first try, seven were obtained on the second try, and one required three tries. Table 4 shows that 38 attempts were made for AMRAD's second tracker; of these, 22 hits were obtained on the first try while the remaining 16 hits required a second attempt. Taken together, we see that in all but one case, a hit was obtained within two attempts. In that remaining case, one additional attempt was required. These results are summarized in Table 5. In all but one case, a solid track was established by HAPDAR subsequent to the first hit after handover. The one time this did not occur, 18 hits and 11 misses were recorded before the next handover attempt.

Although these statistics are quite good, a higher number of handover hits on the first attempt were expected due to the good SNR. However, the lower number of first attempt hits can be explained by the fact that the two radars have not as yet been properly calibrated relative to each other in signal-to-noise sensitivity. Due to the different sensitivities of the two radars, the SNR seen by AMRAD must be properly adjusted to obtain a good estimate of the SNR that HAPDAR should see. This is currently done by subtracting 23 dB from the AMRAD measured SNR before

Table 5

SUMMARY OF ROME SPHERE HANDOVERS

	Number of Handovers	Percentage of Hits on First Pulse	Percentage of Hits on First or Second Pulse
First track	38	78.9%	97.4%
Second track	38	57.9	100
Both tracks	76	68.4	98.7

handing over the AMRAD data to HAPDAR. Should the SNR seen by HAPDAR be consistently lower than predicted by AMRAD at the time of handover, the AGC setting for the first HAPDAR track pulse will be set consistently too low so that the probability of missing a return increases. The probability of missing a return will be greater as the difference between the SNRs becomes greater. When a miss is obtained on the first try, the AGC is increased by steps of 3 dB for subsequent track pulses. Thus, one might expect that if the predicted SNR was generally too high, there might be a significant number of misses on first attempts, and a very small number of misses on second attempts.

To determine whether this occurred for the Rome Sphere track, the handover data were separated into two groups. The first group consisted of data where the difference between the AMRAD predicted SNR and the HAPDAR measured SNR was less than 4 dB. The second group consisted of data where the difference was greater than 4 dB. (The largest difference obtained was 7.1 dB, and the predicted difference was always greater than the measured difference.) The resulting summaries of handover for each group are shown in Table 6. From this table we see a dramatic improvement for the first group, and very poor performance for the second group.

Table 6

SUBSET SUMMARIES OF ROME SPHERE HANDOVERS

	Number of Hits on First Pulse	Number of Hits on Second Pulse	Number of Hits on Third Pulse
SNR difference less than 4 dB			
First track	30	3	0
Second track	14	0	0
Both	44	3	0
SNR difference greater than 4 dB			
First track	0	4	1
Second track	8	16	0
Both	8	20	1

Or, stated another way, we see that 87.5 percent of the misses on the first attempt occurred when the difference between the predicted and measured SNR was greater than 4 dB.

Thus the data seem to indicate very good first pulse handover performance in light of the state of calibration between AMRAD and HAPDAR. The data suggest that the SNR adjustment of the AMRAD measurements handed over to HAPDAR be increased by about 3 dB--from 23 dB to 26 dB.

The data obtained on the handover of the Rome Payload are tabulated in Tables 7 and 8 for the first and the second AMRAD trackers, respectively. The data in these tables indicate a fairly good performance of handover in terms of first hits. However, these data are not considered very reliable because the elevation angle was quite low (about 6°) and the radar target was at a range that is in the region of high clutter.

Table 7

AMRAD FIRST TRACKER HANDOVER DATA FOR ROME PAYLOAD

Attempt Number	Handover Time (seconds)	Predicted SNR (dB)	Time of 1st Hit (seconds)	Measured SNR (dB)	Range Error on 1st Hit (meters)	Number of Misses Before 1st Hit
1	43.40	23.2	43.45	20.4	-19.6	0
2	45.40	17.8	45.55	17.1	-25.8	1
3	47.40	12.2	47.49	17.3	-30.3	0
4	49.40	15.3	49.49	20.9	-23.4	0
5	51.40	18.5	51.49	26.3	-22.9	0
6	53.40	15	53.49	25.2	-16.3	0
7	55.40	--	*	*	*	*
8	57.62	< 10	57.72	19.3	-5.9	0
9	59.62	< 10	59.78	11.9	-6.2	1
10	61.62	11.9	61.70	19.5	-30.3	0
11	63.62	15.3	63.72	25.3	-12.1	0
12	65.72	12.9	66.12	8.4	13.2	5
13	67.62	14.6	67.78	6	-13	1
14	69.62	14.3	69.74	16.4	-21	0
15	--	18.5	74.29	--	29.9	1
16	76.11	12.5	76.15	18.7	1.63	0
17	78.11	13.6	--	--	--	0
18	--	--	80.31	20.6	-28.6	0
19	--	--	82.91	13	-29.6	2
20	84.52	< 10	84.63	17.9	-30.2	0
21	86.62	< 10	86.69	26.6	-29.1	0
22	88.62	< 10	88.67	15.2	-11.6	0
23	90.62	< 10	90.69	18.5	-8.3	0
24	92.62	< 10	92.67	21.4	3.8	0
25	94.72	< 10	94.84	20.9	-13.5	1
26	96.72	< 10	96.78	25	-21	0
27	98.81	< 10	99.08	23.3	-13.5	1

*No hits.

Table 8

AMRAD SECOND TRACKER HANDOVER DATA FOR ROME PAYLOAD

Attempt Number	Handover Time (seconds)	Predicted SNR (dB)	Time of 1st Hit (seconds)	Measured SNR (dB)	Range Error on 1st Hit (meters)	Number of Misses Before 1st Hit
1	44.44	< 10	44.51	16.8	30	0
2	46.44	18.4	46.63	15.3	15.2	1
3	48.44	21.3	48.73	16.1	30.2	3
4	50.44	16.6	50.53	19.5	15.3	0
5	52.44	19.8	52.57	17	30.2	0
6	54.56	17.2	54.67	15.5	30.2	0
7	56.56	--	*	*	*	*
8	58.85	17.9	--	--	--	0
9	60.85	13.4	60.94	22.6	-29.7	0
10	62.85	13.6	62.94	12.7	30.2	0
11	64.85	10.3	64.93	15.7	15.2	0
12	66.85	11.8	66.98	16	-59.8	0
13	71.05	13.9	--	--	--	1
14	--	--	--	--	--	0
15	75.25	13.8	75.33	13.5	30.2	0
16	77.15	12.3	--	--	--	1
17	--	--	--	--	--	0
18	81.25	< 10	81.53	20.9	-89.8	1
19	83.34	< 10	83.49	13.3	-18	0
20	85.44	< 10	85.83	1.36	-21	4
21	87.34	< 10	87.47	11.9	30.2	0
22	89.34	< 10	89.43	14.9	30.2	0
23	91.34	< 10	91.43	15.5	30.2	0
24	93.34	< 10	93.43	10.6	45.2	0
25	95.34	< 10	95.5	21.4	0.09	1
26	--	--	97.48	18.2	15.2	0

*No hits.

This raises the possibility of obtaining an excessive number of hits from clutter rather than from any real target. This type of situation is suggested by the fact that during this period of operation many tracks were initiated and then purged. None of these independent tracks (i.e., non-dedicated tracks) and only a few handed-over tracks were ever solidly established. In addition, the AMRAD predicted SNRs do not appear to correlate very well with the measured SNRs, as can be seen from Tables 7 and 8. On the other hand, for the Rome Sphere data, the SNRs (though somewhat biased) correlated fairly well. In general, AMRAD was predicting a smaller SNR for the Rome Payload than was measured, which is opposite to the trend on the Rome Sphere. In particular, toward the end of the track, AMRAD was consistently predicting SNRs of less than 10 dB while SNRs significantly greater than 10 dB were being measured by HAPDAR.

The conclusion appears to be that little can be said about the performance of handover on the Rome Payload because of the apparent high clutter effects.

The data obtained on the handover of the Rome Rocket are tabulated in Tables 9 and 10 for the first and second AMRAD trackers, respectively. Also tabulated in each of these tables are the number of hits and misses for each of the handed-over tracks, just prior to the next handover attempts. These data indicate that the second AMRAD track was never handed over successfully, while only partial success was achieved on the first AMRAD track handovers. This is consistent with the low SNR indicated by AMRAD on the second track.

The SNRs indicated by AMRAD on the first track are considered marginal in view of the fact that the RCS can be expected to fluctuate. The total number of hits and misses indicated by the handed-over track on the first target appears to be consistent with the SNRs. It is interesting to note that--just as for the Rome Sphere data--the number of attempts

Table 9

AMRAD FIRST TRACKER HANDOVER DATA FOR ROME ROCKET

Attempt Number	Handover Time (seconds)	Predicted SNR (dB)	Time of 1st Hit (seconds)	Measured SNR (dB)	Range Error on 1st Hit (meters)	Number of Misses Before 1st Hit	Hits/Misses Before Next Handover
1	--	--	--	--	--	0	24/4
2	2822.98	16.2	2823.22	10	-21.1	2	21/8
3	2824.98	13.7	2825.26	8.4	-18.7	3	10/18
4	2826.98	< 10	2827.06	13.6	-30.3	0	19/10
5	2828.98	13.5	2829.12	10	-18.7	1	7/21

Table 10

AMRAD SECOND TRACKER HANDOVER DATA FOR ROME ROCKET

Attempt Number	Handover Time (seconds)	Predicted SNR (dB)	Time of 1st Hit (seconds)	Measured SNR (dB)	Range Error on 1st Hit (meters)	Number of Misses Before 1st Hit	Hits/Misses Before Next Handover
1	2821.53	< 10	*	*	*	*	0/31
2	2823.53	< 10	2823.96	9.2	-59.8	5	2/27
3	2825.53	13.7	2825.60	18.3	-74.7	0	6/22
4	2827.53	< 10	2827.68	5.4	-14.8	1	1/28
5	2829.53	< 10	2829.68	1.9	-89.8	1	--

*No hits.

required for the first hit by HAPDAR on a handed-over track is correlated with the difference between the AMRAD predicted SNR and the measured SNR.

The general conclusion on the Rome Rocket handover data is that the SNR for the second target tracked by AMRAD was too low for any success at handover, and the handovers on the first target were partially successful (i.e., only two out of five attempts resulted in solid handed-over tracks) due to a marginal SNR.

D. Correlation Results

The track data obtained from the Rome Sphere provided a fairly good test of the correlation algorithm. During this part of the Rome mission HAPDAR had three active tracks, as expected: the two dedicated handover track channels and an independent track of the sphere. The correlation algorithm is concerned with how each of the two AMRAD tracks correlate with the independent HAPDAR track. Since both AMRAD trackers were on the same sphere, we would expect that both AMRAD tracks would correlate with the independent HAPDAR track. Fortuitously, in view of the meager correlation data obtained to date, this was not the case. It turned out that there was a range bias between the two AMRAD trackers on the order of 7.5 meters (the first tracker was always greater than the second). Thus, although only one object was tracked, two tracks were obtained which in effect correspond to two spheres with a constant separation of about 7.5 meters. The relative range bias between AMRAD and HAPDAR as obtained by comparing the AMRAD first tracker data to the HAPDAR data was only about 0.4 meters. Thus, the two radars were fairly well calibrated in range for the Rome Sphere tracks.

The two AMRAD tracks are based on independent range measurements but completely correlated angle measurements. Thus, the resulting correlation parameters computed by QUECOR for each AMRAD track paired with

the HAPDAR track are partially correlated with each other. As we shall see, the independent range measurements have the dominant effect on the computed Q's (i.e. the correlation parameters*).

The Q's were computed once every second (once every two seconds for each AMRAD track). Each Q computation provides a separate correlation trial. The Q's obtained for the first two segments of the Rome Sphere track are plotted as a function of mission time in Figure 18, and those for the third and final segment are shown in Figure 19. From Figure 18 we see that the Q's for the first AMRAD track and independent HAPDAR track pair range from about 0.7 to 12. On the other hand, the Q's for the second AMRAD track and independent HAPDAR track pair range from 2.6 to 82. From Figure 19 we see that the Q's for the first pair of tracks range from 0.4 to 40 while those for the second pair range from 6 to 143. Although these results indicate a good degree of overlap, the frequency of overlap is low. We see this by considering the frequency of exceedances of some threshold by the first track pair Q's, and the frequency of nonexceedances of the same threshold by the second track pair Q's. If we consider a threshold of 13.0, we see that over all three segments of data (Figures 18 and 19) we have 10.3 percent exceedances for the first track pair Q's, and 11.1 percent nonexceedances for the second track pair Q's. In the context of making correlation decisions, the results indicate that with a correlation decision threshold of 13 and an object separation of only 7.5 meters, the correct decision as to whether the tracked objects were the same or not would have been made a total of 67 out of 75 times or about 89 percent of the time. Four out of the eight erroneous decisions would have been made in designating two

---

*See Chapter III, Section E.4.

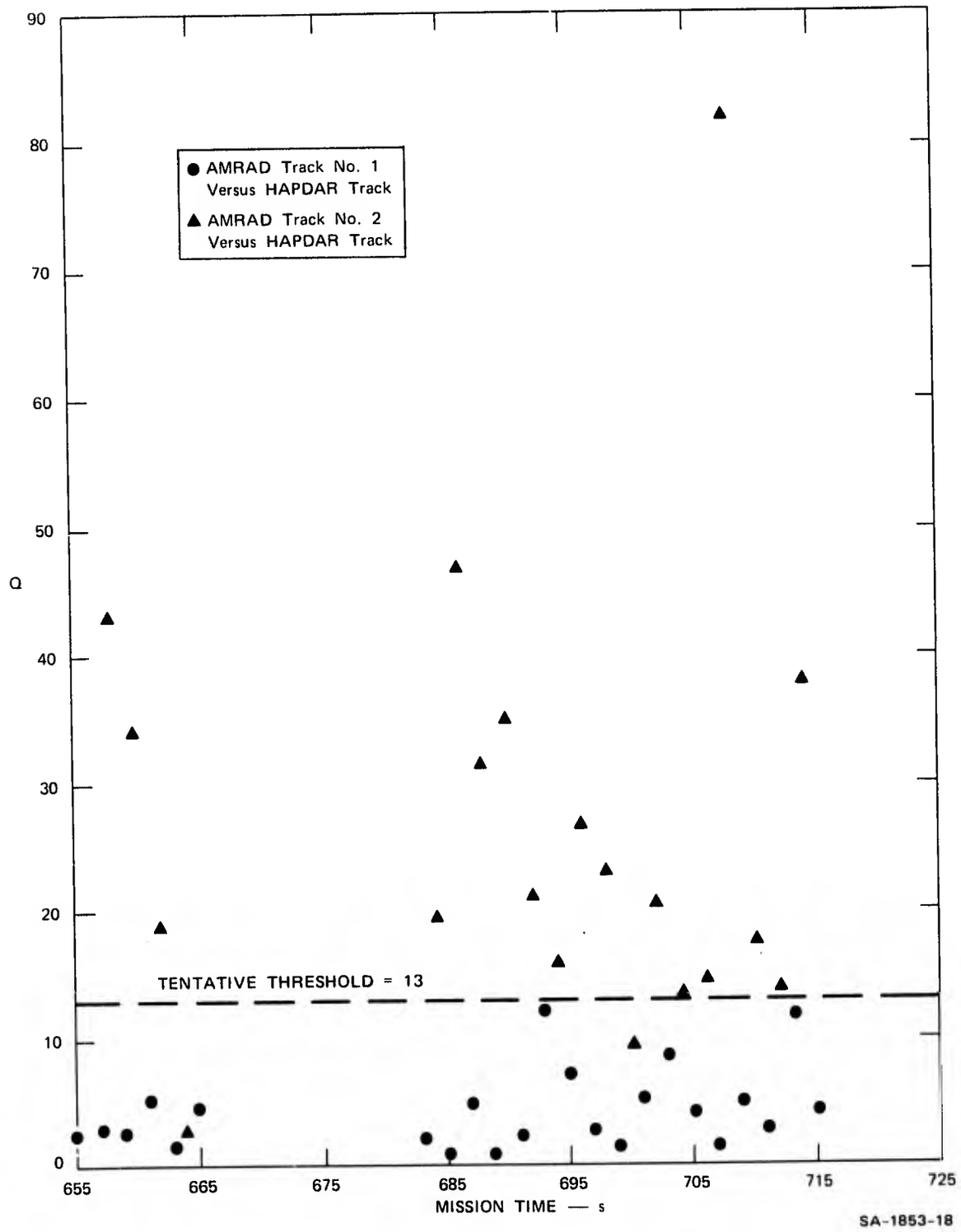


FIGURE 18 ROME SPHERE CORRELATION DATA, PART I

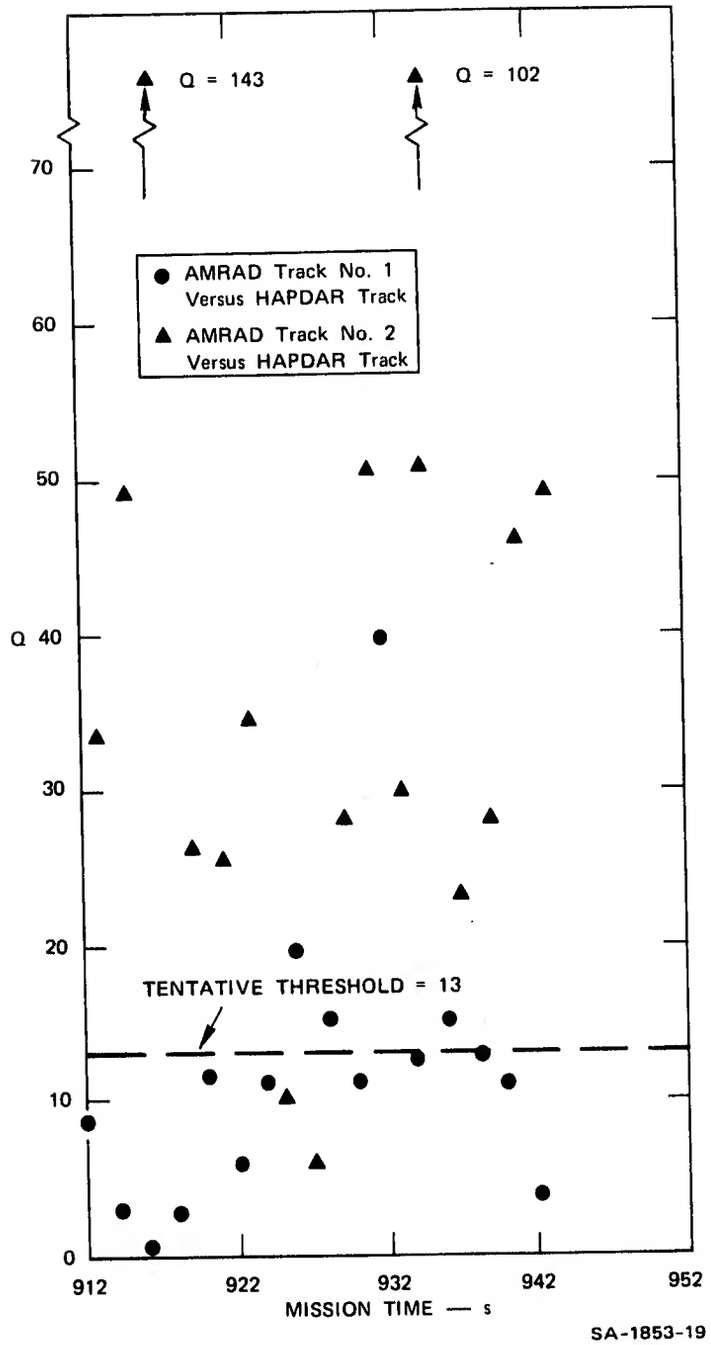


FIGURE 19 ROME SPHERE CORRELATION DATA, PART II

correlated tracks as uncorrelated, and the remaining four erroneous decisions would have been made in designating two uncorrelated tracks as correlated.

The average value of the first track pair Q's was about 7.1, while the average for the second track pair was about 33.7. In a bias free situation we would expect the average value for the first track pair of Q's to be 3.0. In general, much better correlation results were obtained in the first two segments of the Rome Sphere data. These data show average Q's of about 4 and 26. Thus, the expected average Q for the first track pair was almost achieved with the earlier data.

These results indicate a very good performance of the correlation algorithm in view of (1) the fact that angle biases have not yet been calibrated out and (2) the small separation (i.e., the range bias between the AMRAD trackers) of 7.5 meters between the two AMRAD tracks. Such a small separation in the case where there are two real objects would not even result in two separate tracks, since the objects would not be resolved (AMRAD has a range resolution of about 12 meters). In any realistic case, the Q's corresponding to the uncorrelated objects will be much greater than those obtained from the Rome Sphere tracks, and a slightly higher decision threshold would have resulted in almost no incorrect correlation decisions. This last conclusion is supported by the correlation results on the Rome Rocket track data, where there were actually two real objects for AMRAD to track. These results are discussed next.

No correlation results in the sense of correlation of tracks from one radar with independent tracks from a second radar were obtained for the Rome Payload or the Rome Rocket, since no independent tracks were ever established by HAPDAR during those periods of operation. As discussed in the section on handover results this was probably due to high clutter in the case of the Rome Payload, and low SNR coupled with the fact that the

system abnormally terminated a short time after initiation of the handover experiment in the case of the Rome Rocket.

However, some results related to correlation are available because all HAPDAR tracks, including the two dedicated HAPDAR tracks, are processed through QUECOR approximately once every second. These dedicated HAPDAR tracks are not independent of the AMRAD tracks since they are reinitialized with the AMRAD track data once every two seconds for dedicated track. These dedicated tracks are never purged even though no hits are ever obtained. Thus, just after they are reinitialized, or if time passes without any hits by HAPDAR, they should be highly correlated with the AMRAD tracks. However, after a number of hits are obtained they are essentially independent of the AMRAD tracks. The number of hits required for the validity of this last statement is not known precisely, but is believed to be on the order of ten hits.

The unique feature of the Rome Rocket data is that two real targets were tracked by AMRAD, and one of them was handed over with partial success. The first AMRAD track had a higher SNR than the second and was apparently the booster of the Rome Rocket; the second AMRAD track then corresponds to the ejected instrument package payload for the Rome mission. The separation between these two objects was approximately 600 meters. This separation is typical of the separation that might be encountered in a tactical situation. At such separations, we would expect quite large values of Q when we attempt to correlate the two separate objects. The values of the Q's corresponding to the attempts at correlation of each of the AMRAD tracks with the first dedicated track at HAPDAR are tabulated in Table 11.

The numbers in Table 11 indicate a range of Q's of from about 5 to 38 for the pair corresponding to the first AMRAD track. On the other hand, a range of Q's of from about 2000 to 8500 was obtained for the

Table 11

CORRELATION DATA FOR ROME ROCKET

Correlation Time (seconds)	Hits/Misses for SRI-1*	Q Value for AM-1* Compared with SRI-1	Q Value for AM-2* Compared with SRI-1
2821.53	7/0		2500
2822.98	24/4	37.9	
2823.53	4/3		2480
2824.98	21/8	19.6	
2825.53	4/3		8440
2826.98	10/18	5.14	
2827.53	5/2		1980
2828.98	19/10	15.9	
2829.53	6/1		2530

*SRI-1 is used to designate the first HAPDAR dedicated track, AM-1 designates the first AMRAD track, and AM-2 designates the second AMRAD track.

pair corresponding to the second AMRAD track. The higher than expected Q's for the first AMRAD track is due to the fairly large bias in range between the two radars. However, the very large Q's for the uncorrelated tracks (i.e., the second AMRAD track and the first dedicated HAPDAR track) are primarily due to the 600-meter separation between the two real objects.

As indicated in Table 11, by the hit and miss count, QUECOR was consistently executed for the second AMRAD track too soon after the first AMRAD track. Thus, because of the relatively small number of hits we cannot say that the first dedicated HAPDAR track is independent of the

first AMRAD track. However, this does not mean that the magnitudes of the Q's obtained would be much different. It does mean that we would not see the proper statistical variation of the Q's over a large number of such trials.

Thus, we can expect that with separations of several hundreds of meters between objects, large separations in the magnitudes of the computed correlation parameters will result and will allow proper correlation decisions with a high degree of success.

Appendix A

HANDOVER PROGRAM AMTOHAP

Appendix A

HANDOVER PROGRAM AMTOHAP

A. Objective of Handover Program

The purpose of the program AMTOHAP is to take data gathered by the primary and secondary AMRAD trackers and hand the data over to the HAPDAR computer via the MILGO interface and kineplex modems. In the program's operating mode, AMRAD data come directly from the AMRAD radar to the SDS 920 computer; in the test mode, AMRAD's Sigma 5 computer generates a vehicle trajectory and, simulating the radar, sends this data to the SDS 920 computer. (The AMRAD-to-HAPDAR coordinate conversion and vehicle-trajectory smoothing are now performed at HAPDAR's IBM 360/65 computer.) The coordinate handover is accomplished in two steps: the first target and its AGC setting are handed over following reception of a predict interrupt from the MILGO interface, and the second target is handed over following reception of a sync interrupt from the MILGO interface. To allow for post-mission analysis of the handover operation, all pertinent AMRAD and HAPDAR data are recorded on magnetic tape at ten-second intervals by the program AMTOHAP.

B. Description of Program AMTOHAP

The program AMTOHAP is an interrupt-driven algorithm that provides the software connection between the AMRAD and HAPDAR radars. Its historical development, mathematical structure, logical structure, interrupt structure, and operating modes are outlined in turn below. A listing of the program AMTOHAP will be found at the end of this appendix.

1. Historical Developments

The program AMTOHAP was originally developed under the previous contract at SRI on the Institute's CDC 6400 computer. Several versions were originated and tested against AMRAD Athena data to determine CPU demands, memory requirements, and tracking capability. These included a Kalman filter algorithm, a least-square processor, and a Type-II filter. The Kalman filter algorithm was found to consume both excessive CPU time and excessive main storage, as well as suffering instability when processing real radar data. The least-squares algorithm on the other hand, although requiring considerably less CPU and storage, still promised to overburden AMRAD's SDS 920. Experimentation with the Type-II filter indicated that its CPU and memory requirements were tolerable, and that it remained stable under all tracking conditions. Accordingly, this filtering algorithm was selected for use in the field demonstrations at WSMR.

The initial versions of AMTOHAP were written in FORTRAN II, since no bit manipulations or interrupt handling were required during the algorithm evaluation studies described above. However, following selection of the Type-II filter, efforts were made to compress the CDC 6400 program and make it run on SRI's SDS 930 computer. In particular, efforts were made to replace the standard FORTRAN floating-point sine and cosine routines with specially coded fixed-point assembly-language algorithms. By means of these algorithms the speed of trigonometric calculations was increased by approximately an order of magnitude. In addition, the FORTRAN magnetic tape read and write routines were replaced with assembly-language algorithms operating in an interlaced mode. That is, the tape read and write functions were able to proceed in parallel with other CPU operations.

Additional assembly-language efforts were made to (1) check the operating mode of the program, (2) idle the program in anticipation of I/O interrupts, (3) handle the actual I/O interrupts, (4) input AMRAD radar measurements, (5) output HAPDAR position and velocity estimates, (6) strip bits from inputted data words, and (7) pack bits for outputted data words. These procedures were initially written at SRI and subjected to a preliminary debugging on SRI's SDS 930. The software was then transported to WSMR and given a final debugging on AMRAD's SDS 920. Finally, additional efforts were made to speed the execution of the program by eliminating redundant calculations and unnecessary indexing operations.

To adapt AMTOHAP for the present contract work, the coordinate conversion, prediction, and conversion routines were removed from the program. (These routines later became part of the program TRANFT.) In addition the program was altered to accept both primary and secondary target coordinates from AMRAD and to transmit them to HAPDAR. Also, a routine was added to automatically modify the SNR level transmitted to HAPDAR when the AMRAD receiver parametric amplifiers were turned off on strong targets.

2. Mathematical Structure

Because the coordinate conversion, prediction, and correction routines were deleted from AMTOHAP, the mathematical structure of the routine degenerated to scaling the SNR for handover to HAPDAR. All remaining mathematical operations consisted of (1) data collection, storage, and retrieval, (2) bit manipulation, (3) and logical checking.

C. AMTOHAP Flowchart

To speed processing, the program AMTOHAP was designed using in-line, rather than modular, code. Figure A-1 shows the basic layout and flow of the algorithm. The program is first initialized (constants defined and flags set) and then put into an idle loop (described below). On receiving an interrupt, the computer leaves the idle loop and checks to see if the second target coordinates should be sent to HAPDAR. Following this operation, the computer checks to see if a new predictor-corrector cycle should be initiated. If so--that is, if flag LFLG or MFLG is zero--the program reads in the latest AMRAD measurement. The algorithm then outputs the collected data to HAPDAR, and returns to the idle loop.

Every fiftieth input-output cycle, the program AMTOHAP writes the collected data on magnetic tape--unless Breakpoint Switch Three has been thrown on the computer console--for post-mission analysis. The program next checks to see if the mission is completed, and if so (that is, if ISTD = 0) the program rewinds the magnetic tape and dumps its contents on the line printer. Next, the program interrogates the teletypewriter for the constant IGO. This constant tells the computer whether or not to terminate the program AMTOHAP; that is, IGO = 0 means end the processing and IGO $\neq$ 0 means return to the beginning of the program. In the latter case, the magnetic tape is rewound and the program reinitialized. In particular, the contents of the magnetic tape are destroyed.

1. Interrupt Structure

Owing to the presence of five interrupts of varying priority level, some sophistication was required to ensure that the timing and control functions were properly handled in AMTOHAP. During normal operation the program settles into a continuous loop in the subroutine IDLE after completing all required tasks. As shown in Figure A-2, the loop

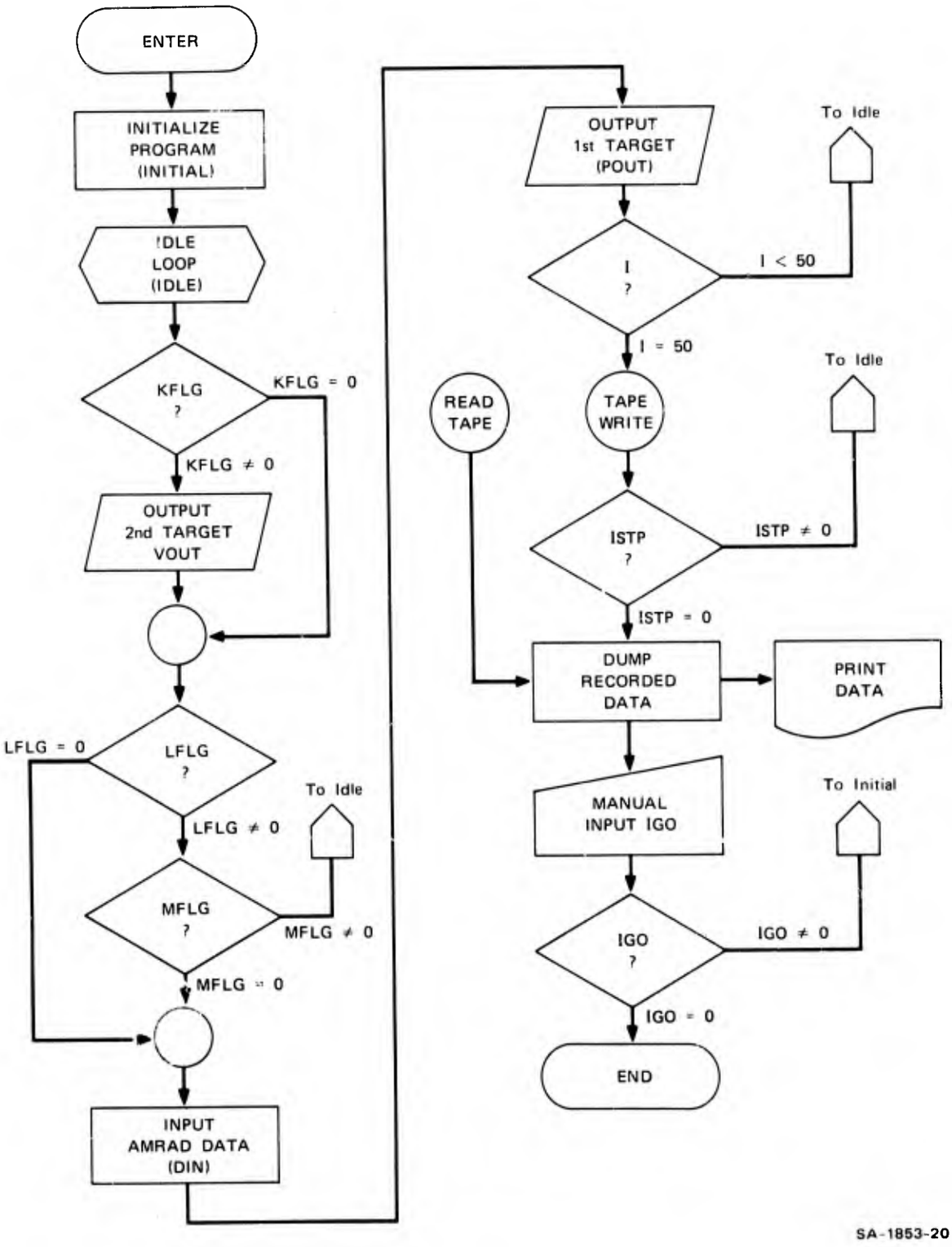


FIGURE A-1 AMTOHAP FLOW CHART

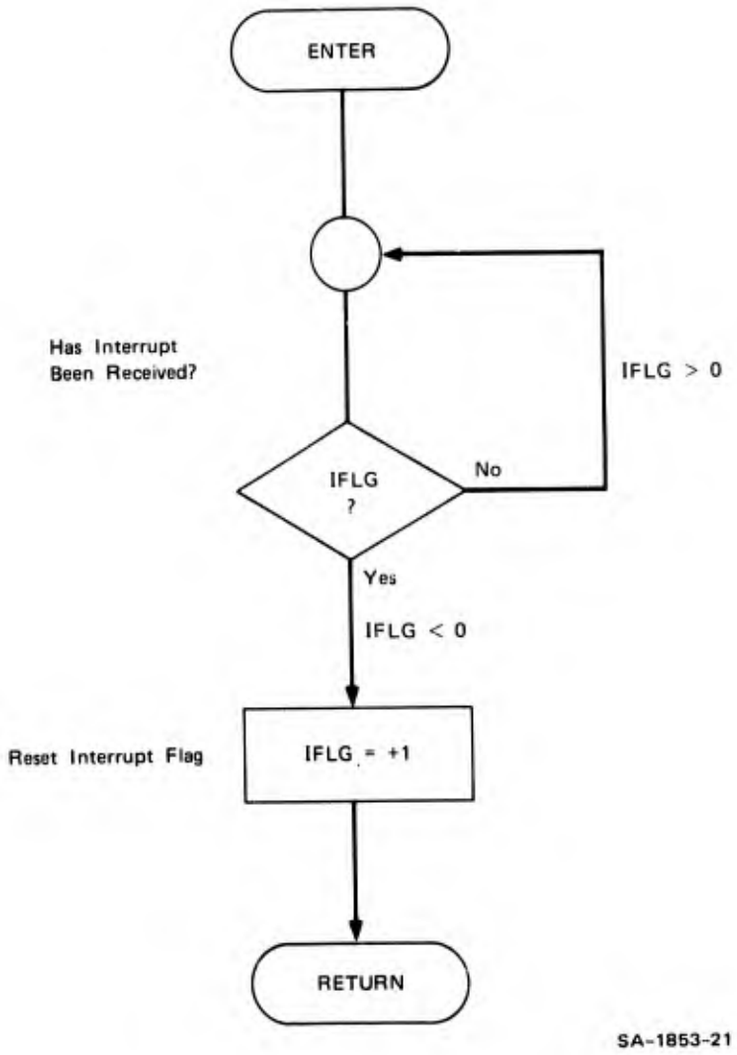


FIGURE A-2 AMTOHAP IDLE LOOP (IDLE)

action depends on the positive value of the interrupt flag IFLG. When an external interrupt appears, the computer stops its present task and executes the interrupt service routine corresponding to that interrupt: that is, SER30, SER31, SER32, SER33, or SER200. The service routines SER31 and SER33 serve to clear the interrupts associated with the tape write and read routines, while the remaining three service routines control the overall handover procedure. Figures A-3, A-4, and A-5 show flow diagrams for the various interrupts.

When an external interrupt causes SER30, SER32, or SER200 to be executed, the interrupt flag IFLG is immediately set equal to -1 by

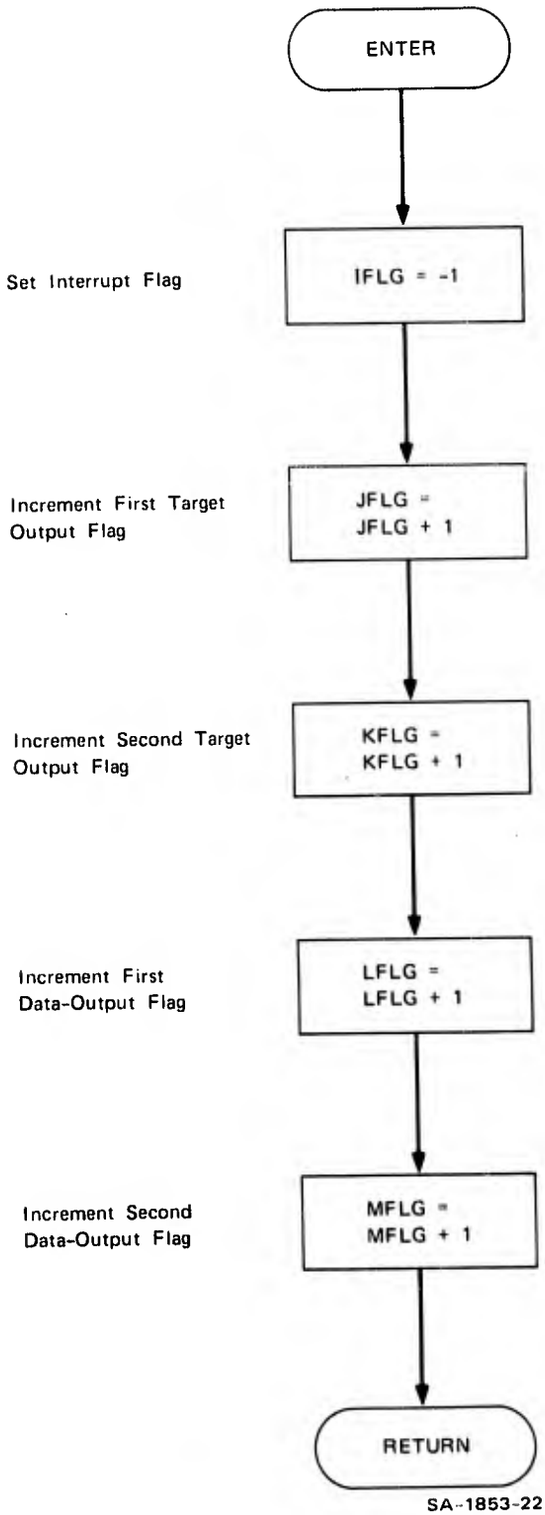


FIGURE A-3 AMRAD MAIN BANG INTERRUPT (SER30)

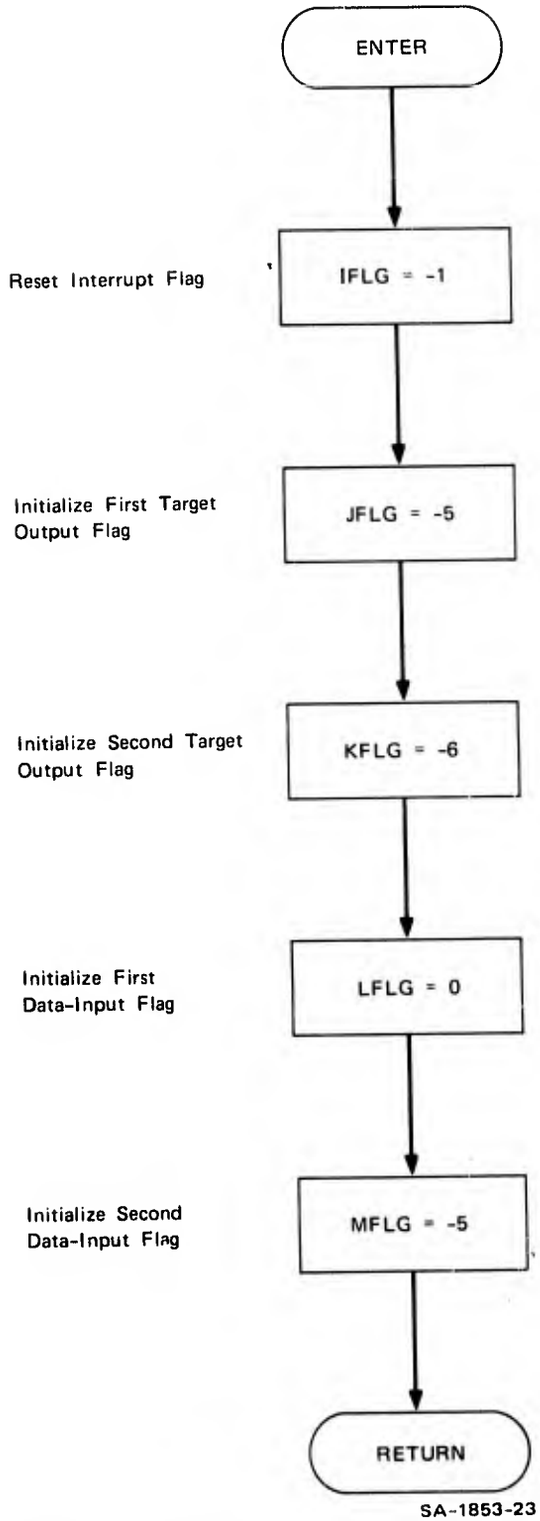


FIGURE A-4 KINEPLEX PREDICT INTERRUPT (SER200)

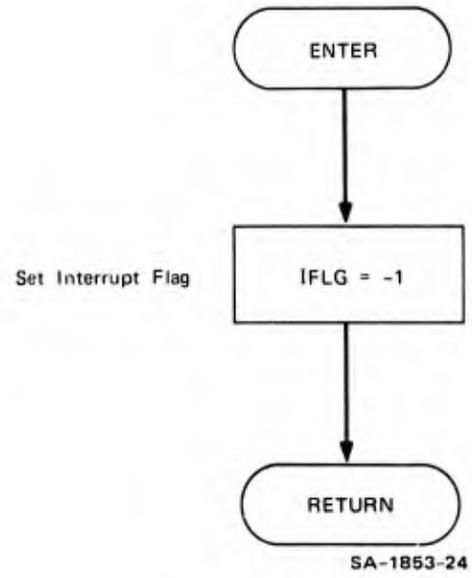


FIGURE A-5 VELOCITY SYNC INTERRUPT (SER32)

the executed routine. Accordingly, on return the computer will immediately jump out of the IDLE loop and go back to the main program. If the second target output flag, KFLG, has been zeroed by a SER30 increment, then the program will immediately output second target data to HAPDAR via the assembly-language subroutine VOUT (see Figure A-1). The computer next checks to see if one of the data-input flags, LFLG or MFLG, has been zeroed by a SER30 increment; if so, new data is immediately read in from the HAPDAR radar, and the resulting position data outputted to the HAPDAR radar by the subroutine POUT. A position-out flag, JFLG, was originally intended to schedule the output time for the position coordinates, but this option was not used for the field experiments.

The kineplex predict and AMRAD main bang interrupts occur "simultaneously." However, because slight timing delays always develop in electronic circuitry, one interrupt will tend to arrive before the

other, and the order of execution of SER32 and SER200 will remain unpredictable. Since SER200 reinitializes the interrupt system and SER32 increments it, these routines must be executed in proper order to ensure correct timing on the handover message. This ordering problem is solved in AMTOHAP by using the higher priority assigned to SER32. Thus, if the AMRAD main bang interrupt arrives first, SER200 will be executed after SER32, and the initialization process follows the incrementing operation. On the other hand, when the kineplex sync interrupt arrives first, the execution of SER200 will begin, but then be interrupted almost immediately by the AMRAD main bang interrupt. Because of priorities, the computer will drop SER200 and begin executing SER32 instead. On completing SER32, it will return again to SER200 and complete its execution. In other words, the initialization process will again follow the incrementing operation.

2. Operating Modes

The program AMTOHAP features six distinct operating modes: normal, printing, recording, dumping, biasing, and Don Site. The operator selects these modes by throwing an appropriate breakpoint switch on the computer console or on an adjacent electronic panel. By throwing more than one such switch, mixed mode operation is obtained.

a. Normal Mode

In the normal operating mode the program AMTOHAP receives data from AMRAD and hands the smoothed data to HAPDAR. No tape recordings are made in this mode and the program is interrupt driven.

b. Printing Mode

When operating in this mode the program AMTOHAP uses the line printer to output data as it is received and calculated. The program is not interrupt driven in this mode, since the line printer limits the speed of execution. This mode is used primarily for debugging and validity checking.

c. Recording Mode

During this mode of operation, the computer operates in the normal mode, except that the collected and calculated data are recorded on magnetic tape.

d. Dumping Mode

The dumping mode is used to transfer the data recorded on magnetic tape to the line printer for direct analysis or permanent storage. On finishing the dump, the program AMTOHAP requests a teletype input to determine whether it should return to the normal mode or get off the computer.

e. Biasing Mode

When operating in this mode a 30-dB bias is added to the measured signal strength to account for attenuation by AMRAD's clutter fence. Otherwise this mode is identical to the normal one.

f. Don Site Mode

The Don Site mode is used to check the calibration of AMRAD and HAPDAR. When the breakpoint switch is thrown, incoming AMRAD data are ignored and replaced instead with the known Don Site coordinates. Otherwise, the program operates in the normal mode.

```
>ENDJOB.
>ASSIGN S=MT0W,SI=CR,LO=LP,BO=PP,BI=PR,X1=MT1W,X2=MT2W,X3=MT3W.
>ASSIGN BO=MT2W.
>ASSIGN BI=MT2W.
>REWIND MT2W.
>FORTRAN SI,BO,LO
C       AMTOHAP -- VERSION II -- SDS 920 AMRAD-TO-HAPDAR HANDOVER PROGRAM
C       CONSOLE BREAKPOINTS -- 2 = PRINT DATA, 3 = RECORD DATA, 4 = DUMP TAPE
C       PANEL BREAKPOINTS -- 9 = SIGNAL BIAS, 10 = DON SITE COORDINATES
C
C       NOTATION -- FIRST CHARACTER = COORDINATE MEASURED, SECOND CHARACTER =
C       MEASURING RADAR, THIRD CHARACTER = TARGET MEASURED.  ALSO, R= RANGE,
C       E = ELEVATION, A = AZIMUTH OR SIN(ALPHA), B = SIN(BETA),  ALSO,
C       A = AMRAD AND H = HAPDAR.  ALSO, SNR = SIGNAL-TO-NOISE RATIO,
C       T = TARGET,.  FOR EXAMPLE, EAT = ELEVATION OF TARGET AS MEASURED FROM
C       AMRAD.  ALSO, S = SINE, C = COSINE.
C       ALSO, V DENOTES SECOND TARGET AND I AN INTEGER QUANTITY.
C
        DIMENSION IDATA(50,13),JDATA(50,13)
C
C       DIMENSIONS - RANGE = METERS, ANGLE = RADIANS, TIME = SECONDS,
C       SIGNAL = DBM AT AMRAD, DBSNR AT HAPDAR
C       INITIALIZE FLAGS AND SET INTERRUPT ADDRESSES
16      REWIND 1
17      IFLG=-100
        JFLG=-100
        KFLG=-100
        LFLG=-100
        MFLG=-100
        IPAR=-1
        CALL SET
C
C       CYCLE FILTER ON INCOMING DATA
18      I=0
        J=0
        K=0
C
C       BASIC FILTERING LOOP BEGINS HERE
20      I=I+1
        J=J+1
C
C       IDLE PROGRAM HERE
22      CALL IDLE(IFLG,JFLG,KFLG,LFLG,MFLG)
C
C       ARM AND ENABLE INTERRUPTS
        CALL INT
C
C       CHECK BREAKPOINT SWITCHES
        CALL CHK(IPRT,IRCD,ISTP)
C
C       OUTPUT SECOND-TARGET COORDINATES TO HAPDAR WHEN JFLG=0 OR KFLG=0
25      IF(JFLG) 26,2025,26
2025    CALL VOUT(IVRAT,IVAAT,IVEAT,IVSAT,IVTAT,ISTAT2,IPUMP)
        GO TO 2027
26      IF(KFLG) 27,2026,27
2026    CALL VOUT(IVRAT,IVAAT,IVEAT,IVSAT,IVTAT,ISTAT2,IPUMP)
C
C       PRINT COLLECTED DATA
2027    IF(IPRT) 27,127,27
127     PRINT 15,IRAT,IAAT,IEAT,ISAT,ITAT,ISTAT1,IPUMP
        PRINT 15,IVRAT,IVAAT,IVEAT,IVSAT,IVTAT,ISTAT2,IPUMP
        PRINT 15,IFLG,JFLG,KFLG,LFLG,MFLG
        PRINT 15
```

```
        BRR     TAPE
IDATA   RES     2
NWRD    EQU     650
ITAPE   EQU     1
WRITE   VFD     650,0
        PAGE
P       PROC                    DEFINE BREAKPOINT PROC FOR PANEL SWITCHES
BPNT    NAME    040
I       FORM    3,6,6,9
*
        DO      P(1)-5,1,2
        BPT     P(1)
*
        I       0,040,030,2**/(P(1)-5)
        END
        END
>EOF
>WEOF MT2W.
>ENDJOB.
>REWIND MT2W.
>ASSIGN L1=MT1W.
>REWIND MT1W.
>FORTLOAD MAP,LGO,BI.
>ENDJOB.
```

```
          PAGE
$CHK      PZE                         CHECK SETTINGS OF BREAKPOINT SWITCHES
          BRM       201SYS
          XDS       IPRT
          XDS       IRCD
          XDS       ISTP
          BRM       202SYS
          BPT       2                 SHALL WE PRINT DATA
          BRU       $+4
          LDA       =1
          STA       *IPRT
          BRU       $+3
          LDA       =0
          STA       *IPRT
          BPT       3                 SHALL WE RECORD DATA
          BRU       $+4
          LDA       =1
          STA       *IRCD
          BRU       $+3
          LDA       =0
          STA       *IRCD
          BPT       4                 SHALL WE STOP TRANSMITTING
          BRU       $+4
          LDA       =1
          STA       *ISTP
          BRR       CHK
          LDA       =0
          STA       *ISTP
          BRR       CHK
IPRT      RES       2
IRCD      RES       2
ISTP      RES       2
          PAGE
$GET      PZE                         GET RECORDED DATA OFF TAPE WITH INTERLACE
          BRM       201SYS
          XDS       IDATA
          BRM       202SYS
          DIR
          TRT       *,ITAPE
          BRU       $+2
          BRU       $-2
          LDA       READ
          ETR       =077740000
          MRG       IDATA
          STA       READ
          RTB       *0,ITAPE,4
          POT       READ
          TRT       *,ITAPE
          BRU       $+2
          BRU       $-2
          BRR       GET
READ      VFD       60/0
          PAGE
$TAPE     PZE                 WRITE COLLECTED DATA ON TAPE WITH INTERLACE
          BRM       201SYS
          XDS       IDATA
          BRM       202SYS
          TRT       *,ITAPE
          BRU       $+2
          BRU       $-2
          LDA       WRITE
          ETR       =077740000
          MRG       IDATA
          STA       WRITE
          WTB       *0,ITAPE,4
          POT       WRITE
```

```
        XDS     IVEAT
        XDS     IVSAT
        XDS     IVAT
        XDS     ISTAT2
        XDS     IPUMP
        BRM     202SYS
        BPNT    10              SHALL WE SEND DON SITE COORDINATES
        BRU     VVV
        LDA     DON
        STA     *IVRAT
        LDA     DON+1
        STA     *IVEAT
        LDA     DON+2
        STA     *IVAAT
        LDA     DON+3
        STA     *IVSAT
*
VVV     LDA     *IVRAT
        ETR     =03777777
        MRG     =014000000
        STA     *IVRAT
*
        LDA     *IVAAT
        ETR     =0377777        EXTRACT RIGHT 17 BITS.
        STA     *IVAAT
*
        LDA     *IVEAT
        ETR     =0377777        EXTRACT RIGHT 17 BITS.
        STA     *IVEAT
*
        LDA     *IVTAT
        CLB                     CLEAR B
        LCY     18              PUT LEAST 6 BITS IN LEFT 6 BITS OF A.
        MRG     *IVEAT          MERGE WITH ELEVATION
        STA     *IVEAT          STORE IN ELEVATION
        CBA                     COPY B REGISTER TO A .
        CLB                     CLEAR B
        LCY     18              PUTS MOST 6 BITS IN LEFT 6 BITS OF A.
        MRG     *IVAAT
        STA     *IVAAT          STORE IN AZIMUTH
*
        LDA     *IVSAT           LOAD A REG WITH SIX-BIT SIGNAL
        CLB                     CLEAR B REG
        LCY     2               LEFT CYCLE AB BY 2 BITS
        MRG     *ISTAT2         MERGE STATUS WITH SIGNAL
        CLB                     CLEAR B REG
        LCY     3               LEFT CYCLE AB BY THREE BITS
        ADD     =01             ADD ONE TO MERGED SIGNAL
        CLB                     CLEAR B REG
        LCY     13              LEFT CYCLE AB REG BY 13 BITS
        STA     *IVSAT           STORE RESULT IN IVSAT
*
        EOM     030041
        POT     *IVRAT
        EOM     030220
        POT     *IVAAT
        EOM     030201
        POT     *IVEAT
        EOM     030021
        POT     *IVSAT
        BRR     VOUT
IVRAT   RES     2
IVAAT   RES     2
IVEAT   RES     2
IVSAT   RES     2
IVTAT   RES     2
```

```
        XDS     ISAT
        XDS     ITAT
        XDS     ISTAT1
        XDS     IPUMP
        BRM     202SYS
        BPNT    10              SHALL WE SEND DONSITE COORDINATES
        BRU     PPP
        LDA     DON
        STA     *IRAT
        LDA     DON+1
        STA     *IEAT
        LDA     DON+2
        STA     *IAAT
        LDA     DON+3
        STA     *ISAT
*
PPP     LDA     *IRAT
        ETR     =03777777       EXTRACT RIGHT 20 BITS OF RANGE.
        MRG     =010000000      MERGE WITH ID BITS
        STA     *IRAT
*
        LDA     *IAAT
        ETR     =0377777        EXTRACT RIGHT 17 BITS OF AZIMUTH
        STA     *IAAT
*
        LDA     *IEAT
        ETR     =0377777        EXTRACT RIGHT 17 BITS OF ELEVATION
        STA     *IEAT
*
        LDA     *ITAT           LOAD A WITH 12-BIT TIME
        CLB                     CLEAR B
        LCY     18              PUT LEAST 6 BITS IN LEFT 6 BITS OF A.
        MRG     *IEAT           MERGE WITH ELEVATION
        STA     *IEAT           STORE IN ELEVATION
        CBA                     COPY B REGISTER TO A .
        CLB                     CLEAR B
        LCY     18              PUTS MOST 6 BITS IN LEFT 6 BITS OF A.
        MRG     *IAAT           MERGE WITH AZIMUTH
        STA     *IAAT           STORE IN AZIMUTH
*
        LDA     *ISAT           LOAD A REG WITH SIX-BIT SIGNAL
        CLB                     CLEAR B REG
        LCY     2               LEFT CYCLE AB BY 2 BITS
        MRG     *ISTAT1         MERGE STATUS WITH SIGNAL
        CLB                     CLEAR B REG
        LCY     3               LEFT CYCLE AB BY THREE BITS
        ADD     =01
        CLB                     CLEAR B REG
        LCY     13              LEFT CYCLE AB REG BY 13 BITS
        STA     *ISAT           STORE RESULT IN ISAT
*
        EOM     030041
        POT     *IRAT
        EOM     030220
        POT     *IAAT
        EOM     030201
        POT     *IEAT
        EOM     030021
        POT     *ISAT
        BRR     POUT
DON     DATA    025713,0340366,0364016,077
        PAGE
$VOUT   PZE                     OUTPUT RAPDAR VELOCITY DATA
        BRM     201SYS
        XDS     IVRAT
        XDS     IVAAT
```

```
        RSH     4               DIVIDE BY TWO TO GET ONE DBM/BIT
        ETR     =077            EXTRACT RIGHT SIX BITS
        BPNT    9               CLUTTER FENCE CHECK
        BRU     $+5
        ADD     =30             ADD 30 DB FOR CLUTTER FENCE
        SKA     =0100           CHECK FOR OVERFLOW
        BRU     $+2
        LDA     =077            PLACE CEILING ON AMRAD SIGNAL STRENGTH
        STA     *IVSAT          STORE SECOND SIGNAL IN MEMORY
        LDA     *ISAT           LOAD SIGNAL INTO A REGISTER
        RSH     12              DIVIDE BY TWO TO GET ONE DBM/BIT
        ETR     =077            EXTRACT RIGHT SIX BITS
        BPNT    9               CLUTTER FENCE CHECK
        BRU     $+5
        ADD     =30             ADD 30 DB FOR CLUTTER FENCE
        SKA     =0100           CHECK FOR OVERFLOW
        BRU     $+2
        LDA     =077            PLACE CEILING ON AMRAD SIGNAL STRENGTH
        STA     *ISAT           STORE FIRST SIGNAL IN MEMORY
*
        LDA     *ITAT           PUT TIME IN A
        CLB
        RCY     12
        MRG     *ITIME
        STA     *ITIME
        LDA     *ITAT
        RCY     2
        ETR     =01777          EXTRACT BITS 10-23
        STA     SAVE            SAVE A IN SAVE
        LDA     *ITAT           PUT TIME IN A
        RCY     12              RIGHT CYCLE AB BY 12 BITS
        ETR     =07777          EXTRACT LAST 12 BITS
        MUL     =500            MULTIPLY A BY 1000.
        CBA                     COPY B TO A
        ADD     SAVE            ADD SAVE TO A
        STA     *ITAT           PUT A IN *ITAT
*
        LDA     *IVRAT          LOAD SECOND RANGE INTO A REGISTER
        ETR     =017            EXTRACT LAST 4 BITS
        STA     SAVE            PUT LAST 4 BITS IN SAVE
        LDA     *IVRAT          LOAD A REGISTER WITH SECOND RANGE
        RCY     4               RIGHT CYCLE A REG BY 4 BITS
        ETR     =0777777        EXTRACT LAST 18 BITS
        MUL     =5              MULTIPLY CONTENTS OF A BY 10.
        CBA                     COPY B TO A
        ADD     SAVE            ADD SAVE TO A
        STA     *IVRAT          PUT A IN *IVRAT.
        BRR     DIN
IRAT    RES     2
IEAT    RES     2
IAAT    RES     2
ITAT    RES     2
ISAT    RES     2
IVRAT   RES     2
IVSAT   RES     2
ISTAT1  RES     2
ISTAT2  RES     2
IPUMP   RES     2
ITIME   RES     2
SAVE    RES     1
        PAGE
$POUT   PZE                     OUTPUT RADAR POSITION DATA
        BRM     201SYS
        XDS     IRAT
        XDS     IAAT
        XDS     IEAT
```

```
         PAGE
$DIN     PZE                        BRING AMRAD DATA IN
         BRM       201SYS
         XDS       IRAT
         XDS       IAAT
         XDS       IEAT
         XDS       ISAT
         XDS       ITAT
         XDS       IVRAT
         XDS       IVSAT
         XDS       ISTAT1
         XDS       ISTAT2
         XDS       IPUMP
         XDS       ITIME
         BRM       202SYS
         EOM       034004
         PIN       *IRAT
         EOM       030600
         PIN       *IEAT
         EOM       030300
         PIN       *IAAT
         EOM       034002
         PIN       *ISAT
         EOM       030402
         PIN       *ITAT
         EOM       031010
         PIN       *IVRAT
         LDA       *IRAT            LOAD RANGE INTO A REGISTER
         ETR       =017             EXTRACT LAST 4 BITS
         STA       SAVE             PUT LAST 4 BITS IN SAVE
         LDA       *IRAT            LOAD A REGISTER *ITH RANGE
         RCY       4                RIGHT CYCLE A REG BY 4 BITS
         ETR       =0777777         EXTRACT LAST 18 BITS
         MUL       =5               MULTIPLY CONTENTS OF A BY 10.
         CHA                        COPY B TO A
         ADD       SAVE             ADD SAVE TO A
         STA       *IRAT            PUT A IN *IRAT.
*
         LDA       *IEAT            PUT ELEVATION IN A
         ETR       =037             EXTRACT RIGHT 5 BITS
         CLH
         LCY       12               LEFT CYCLE AB BY 12 BITS
         STA       *ITIME
         LDA       *IEAT
         RCY       5                RIGHT CYCLE AB BY SIXBITS.
         ETR       =0377777         EXTRACT RIGHT 17 BITS.
         STA       *IEAT            STORE ELEVATION
*
         LDA       *IAAT            PUT AZIMUTH IN A
         RCY       1                RIGHT CYCLE AB BY ONE BIT.
         ETR       =03              EXTRACT STATUS OF SECOND TARGET
         STA       *ISTAT2          STORE STATUS OF SECOND TARGET
         LDA       *IAAT            PUT AZIMUTH IN A
         RCY       3                RIGHT AB BY  THREE BITS
         ETR       =01              EXTRACT PUMP SETTING
         STA       *IPUMP           STORE PUMP SETTING
         LDA       *IAAT            PUT AZIMUTH IN A
         RCY       4                RIGHT CYCLE AB BY FOUR BITS
         ETR       =03              EXRTACT STATUS OF FIRST TARGET
         STA       *ISTAT1          STORE STATUS OF FIRST TARGET
         LDA       *IAAT            PUT AZIMUTH IN A
         RCY       6                RIGHT CYCLE AB BY SIXBITS
         ETR       =0377777         EXTRACT 17 BITS OF AZIMUTH
         STA       *IAAT            STORE AZIMUTH
*
         LDA       *ISAT            LOAD SIGNAL INTO A REGISTER
```

```
ERASE   VFD     500,0
        PAGE
$INT    PZE                     ARM AND ENABLE INTERRUPTS
        AIR
        POT     LOC
        EIR
        BRR     INT
LOC     DATA    000300000
        PAGE
$IDLE   PZE                     IDLE LOOP
        BRM     201SYS
        XDS     IFLG            INTERRUPT RECEIVED FLAG
        XDS     JFLG            OUTPUT FIRST SCEOND-TARGET COORDINATES
        XDS     KFLG            OUTPUT SECOND SECOND-TARGET COORDINATES
        XDS     LFLG            INPUT/OUTPUT FIRST DATA SET FROM AMRAD
        XDS     MFLG            INPUT/OUTPUT SECOND DATA SET FROM AMRAD
        BRM     202SYS
        SKN     *IFLG
        BRU     $-1
        LDA     =1
        STA     *IFLG
        BRR     IDLE
IFLG    RES     2
JFLG    RES     2
KFLG    RES     2
LFLG    RES     2
MFLG    RES     2
AREG    RES     2
        PAGE
$SER30  PZE                     SERVICE AMRAD MAINBANG INTERRUPT
        STA     REG30
        LDA     =-1
        STA     *IFLG           RESET INTERRUPT FLAG
        LDA     REG30
        MIN     *JFLG
        MIN     *KFLG
        MIN     *LFLG
        MIN     *MFLG
        BRU     *SER30
REG30   RES     1
$SER31  PZE                     CLEAR READ/WRITE INTERRUPTS
        BRU     *SER31
$SER32  PZE                     SERVICE VELOCITY SYNC INTERRUPT
        STA     REG32
        LDA     =-1
        STA     *IFLG           RESET INTERRUPT FLAG
        LDA     REG32
        BRU     *SER32
REG32   RES     1
$SER33  PZE                     CLEAR READ/WRITE INTERRUPTS
        BRU     *SER33
$SER200 PZE                     SERVICE PREDICT INTERRUPT
        STA     REG200
        LDA     =-1
        STA     *IFLG           RESET MAINBANG INTERRUPT FLAG
        LDA     =-1
        STA     *JFLG           RESET FIRST SECOND-TARGET OUTPUT FLAG
        LDA     =-6
        STA     *KFLG           RESET SECOND SECOND-TARGET OUTPUT FLAG
        LDA     =0
        STA     *LFLG           RESET FIRST-DATA-SET INPUT/OUTPUT FLAG
        LDA     =-5
        STA     *MFLG           RESET SECOND-DATA-SET INPUT/OUTPUT FLAG
        LDA     REG200
        BRU     *SER200
REG200  RES     1
```

```
93      CALL INT
C
C       RECORD COLLECTED DATA ON TAPE 1 AFTER FIFTY CYCLES.
90      IF(IRCD) 100,195,100
195     IF(I-50) 105,95,95
95      IF(IPAR) 196,196,197
196     CALL TAPE(IDATA)
        IPAR=+1
        GO TO 99
197     CALL TAPE(JDATA)
        IPAR=-1
C
99      K=K+1
100     I=0
105     IF(ISTP) 20,200,20
C
C       DUMP COLLECTED DATA ON PRINTER
200     END FILE 1
        REWIND 1
        DO 250 I=1,K
        CALL CHK(IPRT,IRCD,ISTP)
        IF(ISTP) 275,220,275
220     CALL GET(IDATA)
        PRINT 15
250     PRINT 15,((IDATA(II,JJ),JJ=1,13),II=1,50)
275     REWIND 1
C
C       STAY ON OR GET OFF MACHINE
        ACCEPT 300,IGO
300     FORMAT(I1)
        IF(IGO)16,305,16
305     CONTINUE
        END
>EOF
>META920 ST,LO,GO
XDS     OPD     =010000000
VFD     FORM    18,14
$SET    PZE                     DEFINE INTERRUPT ADDRESSES AND SPACE TAPE
        TRT     0,0
        BRU     $+2
        BRU     $-2
        REW     0,0
        TRT     0,ITAPE
        BRU     $+2
        BRU     $-2
        BTT     0,ITAPE
        BRU     $+2
        BRU     $+3
        EFT     *0,ITAPE,4
        POT     ERASE
        LDA     DEF30
        STA     30
        LDA     DEF31
        STA     31
        LDA     DEF32
        STA     32
        LDA     DEF33
        STA     33
        LDA     DEF200
        STA     200
        BRM     SET
DEF31   BRM     SER31
DEF33   BRM     SER33
DEF30   BRM     SER30
DEF32   BRM     SER32
DEF200  BRM     SER200
```

```
15    FORMAT(1H ,13010)
      CALL INT
C
C     RECORD SECOND-TRACK DATA
      IF(IPAR) 31,31,33
31    IDATA(I, 9)=IVRAT
      IDATA(I,10)=IVAAT
      IDATA(I,11)=IVEAT
      IDATA(I,12)=IVSAT
      IDATA(I,13)=ITIME
C     IDATA(I,13)=64*64*KFLG+64*LFLG+MFLG
      GO TO 27
33    JDATA(I, 9)=IVRAT
      JDATA(I,10)=IVAAT
      JDATA(I,11)=IVEAT
      JDATA(I,12)=IVSAT
      JDATA(I,13)=ITIME
C     JDATA(I,13)=64*64*KFLG+64*LFLG+MFLG
C
C     READ IN AMRAD DATA IF LFLG=0 OR MFLG=0.
27    IF(LFLG) 29,32,29
29    IF(MFLG) 22,32,22
C
C     READ IN AMRAD DATA, SAVE IN DATA, AND CONVERT TO METERS.
32    CALL DIN(IRAT,IAAT,IEAT,ISAT,ITAT,IVRAT,IVSAT,ISTAT1,ISTAT2,IPUMP,
     1ITIME)
      IVAAT=IAAT
      IVEAT=IEAT
      IVTAT=ITAT
C     USING DOUBLE BUFFERING BECAUSE OF INTERLACED TAPE WRITE.
      IF(IPAR) 43,43,44
43    IDATA(I,1)=IRAT
      IDATA(I,2)=IAAT
      IDATA(I,3)=IEAT
      IDATA(I,4)=ISAT
      IDATA(I,4)=IVRAT
C     IDATA(I,4)=ISAT+64*IVSAT+4096*ISTAT1+8*4096*ISTAT2+64*4096*IPUMP
      GO TO 45
44    JDATA(I,1)=IRAT
      JDATA(I,2)=IAAT
      JDATA(I,3)=IEAT
      JDATA(I,4)=IVRAT
C     JDATA(I,4)=ISAT+64*IVSAT+4096*ISTAT1+8*4096*ISTAT2+64*4096*IPUMP
C
C     SAVE HAPDAR PREDICTIONS
C
C     OUTPUT FIRST-TARGET COORDINATES TO HAPDAR.
118   CALL POUT(IRAT,IAAT,IEAT,ISAT,ITAT,ISTAT1,IPUMP)
C
      IF(IPAR) 121,121,122
121   IDATA(I,5)=IRAT
      IDATA(I,6)=IAAT
      IDATA(I,7)=IEAT
      IDATA(I,8)=ISAT
      GO TO 123
122   JDATA(I,5)=IRAT
      JDATA(I,6)=IAAT
      JDATA(I,7)=IEAT
      JDATA(I,8)=ISAT
C
C     PRINT COLLECTED DATA
123   IF(IPRT) 90,97,90
97    IF(IPAR) 91,91,92
91    PRINT 15,  (IDATA(I,KK),KK=1,13)
      GO TO 93
92    PRINT 15,  (JDATA(I,KK),KK=1,13)
```

Appendix B

FITTING PROGRAM TRANFT

Appendix B

FITTING PROGRAM TRANFT

This section of the report describes the IBM 360/65 fitting program TRANFT as well as the various methods used to validate its operation.

A. Objective of Fitting Program

The purpose of the program TRANFT is to take data received from AMRAD, convert them to HAPDAR coordinates, smooth them with a tracking filter, and then give the fitted data to the HAPDAR operating system. The RCA routine VPAS calls TRANFT into operation on receipt of new data from AMRAD. TRANFT in turn calls the SRI routine QUECOR, after fitting the new data. The AMRAD-to-HAPDAR coordinate conversion and trajectory smoothing are performed within TRANFT by algorithms developed at SRI.

B. Description by the Program TRANFT

The program TRANFT is an interrupt-driven algorithm that provides track initialization information to HAPDAR from AMRAD. Its historical development, mathematical structure, and logical structure are outlined in turn below.

1. Historical Development

The program TRANFT evolved from software developed in earlier contract work. Several tracking filters were designed and tested against AMRAD Athena data to determine CPU demands, memory requirements, and tracking ability. These included a Kalman filter, a least-square

processor, and a Type-II filter. The Kalman filter algorithm was found to consume both excessive CPU time and excessive main storage, as well as suffering instability when processing real radar data. The least-squares algorithm on the other hand, although requiring considerably less CPU and storage, still required too much fast memory. Experimentation with the Type-II filter indicated that its CPU and memory requirements were tolerable, and that it remained stable under all tracking conditions. Accordingly, this filtering algorithm was selected for use in the field demonstrations at WSMR.

During an early phase of the contract work, an adaptive Type-II filter was designed and fitted. The purpose of this algorithm was to vary the filtering constants according to the amount of vehicle acceleration detected. That is, large constants are derived when the vehicle is accelerating to suppress acceleration lag; similarly, small constants are derived when the vehicle is coasting to obtain well-smoothed coordinate estimates. The algorithm proved very effective in tests at SRI on Athena data, but tended to use excessive computer time. Accordingly, the concept of variable filter constants was dropped in the work that followed.

In addition to the coordinate estimating code obtained from the original version of AMTOHAT, algorithms were needed to assess the errors associated with the position, velocity, acceleration, and SNR estimates. These algorithms were obtained by extending the Type-II filter theory. That is, the predicted coordinates were compared with the measured ones and the differences used to estimate the errors in the filtered values. (The details of the technique are described in more detail below.)

2. Mathematical Structure

The tracking filter built into TRANFT was designed especially for WSMR field demonstrations. The tracking algorithm uses an optionally adaptive Type II filter to obtain its position and velocity estimates, and

an optionally adaptive Type I filter for its acceleration and SNR estimates. This filter is considered superior to both the Type-II and Kalman algorithms for the stated field applications.

a. Filter Design Requirements

In designing the TRANFT filter the following requirements were imposed:

- The filter should process the incoming data recursively to conserve fast memory and central processor time.
- The filter should generate a vehicle state vector and its associated error covariance matrix.
- The filter should adapt dynamically to tracking errors, mathematical nonlinearities, plant noise, and target fading.

To conserve computer resources and make optimum use of the radar measurements, the trajectory filter should operate recursively on the input data. That is, the new trajectory estimate should be obtained by modifying the old estimate using the latest radar measurement. This technique contrasts with the nonrecursive procedure wherein the oldest measurement is dropped in favor of the new one and trajectory reestimated with the fitting algorithm. Unlike the recursive technique, the latter procedure fails to take advantage of previous mathematical work, and thus unnecessarily burdens the digital computer facilities.

To accomplish handover and correlation, the trajectory filter should provide estimates of both the vehicle's state vector and its error-covariance matrix. For the purposes of reentry vehicle tracking a six-component position/velocity state vector has proven adequate. To perform the required target correlation one also needs the associated 6×6 error-covariance matrix. When predicting ahead for handover or

adapting the filter to high acceleration vehicles, a nine-component position/velocity/acceleration state vector is preferred.

Experience indicates that field demonstrations encounter such problems as target jumping by the AMRAD tracker, nonlinearities in the handover coordinate transformation, plant noise in the vehicle's equations of motion, and signal fading due to target motion. Accordingly, it is desirable to have the trajectory filter adapt and recover from such errors as they occur. Toward this end the filter should have the capability to:

(1) Weight the incoming data more heavily than the old data to suppress cumulative type mathematical errors; cumulative type fitting errors; target jumping errors; and errors due to irregularities in the vehicle's equations of motion, inaccuracies in the atmospheric densities, and nonlinearities in the handover coordinate transformation.

(2) Adjust the magnitude of the filtering constants according to the recorded signal-to-noise ratio in order (a) to optimize the use of the incoming data and (b) to coast the vehicle's state vector through periods of target fading.

(3) Adjust the magnitudes of the filtering constants according to the vehicle acceleration in order to reduce tracking lag during periods of high vehicle acceleration.

(4) Derive the error-covariance matrix from the measured errors--rather than the measured SNR--so that the error assignments remain tied to radar measurements.

b. Definition of Symbols

For convenience the various mathematical symbols used herein are defined below:

Z_p = predicted AMRAD state vector

Z_e = estimated AMRAD state vector

Z_m = measured AMRAD state vector

E = error-covariance matrix

F = transition matrix

C = error-correction matrix

H = coordinate-conversion matrix

t = time at which measurement, prediction, or estimate is valid.

The state vector, Z, is columnar and has the following ten-component form:

$$Z^T = (PR, PE, PA, VR, VE, VA, LR, LE, LA, SNR)$$

where P = position coordinate, V = velocity coordinate, and L = lag coordinate. Also, R = range, E = elevation, A = azimuth, and SNR = signal-to-noise ratio. (In practice, an eleventh component, the time t, might be added to complete the state vector Z.)

c. Predicting Ahead

Given the estimated state vector, Z_e, and the estimated error matrix, E_e, the trajectory filter predicts ahead one sampling interval Δt using the transition matrix F:

$$Z_p(t + \Delta t) = FZ_e(t)$$

$$E_p(t + \Delta t) = FE_e(t + \Delta t)F^T \quad .$$

Basically, the transition matrix predicts the new vehicle position by adding a velocity increment to the old position estimate. (For handover an acceleration increment is also added.) To keep the filtering stable, the velocity and acceleration are not modified by the transition matrix F. That is, the velocity and acceleration predicted for the time $t + \Delta t$ equal the velocity and acceleration estimated at the time t.

d. Making Corrections

The tracking filter next corrects the predicted state vector, using the new AMRAD measurements:

$$Z_e = Z_p + C(Z_m - HZ_p) \quad .$$

The velocity coordinates are, of course, not measured by AMRAD. Accordingly, the corresponding elements of Z_m--but not Z_p--are set equal to zero. (The error-correction matrix C is described in detail in Section e below.)

The coordinate-conversion matrix H serves to zero the unmeasured components of the state vector Z_p. Accordingly, it has the following form:

$$\begin{bmatrix} 1 & & & & & & & & \\ & 1 & & & & & & & \\ & & 1 & & & & & & \\ & & & 0 & & & & & \\ & & & & 0 & & & & \\ & & & & & 0 & & & \\ & & & & & & 1 & & \\ 0 & & & & & & & 1 & \\ & & & & & & & & 1 \\ & & & & & & & & & 1 \end{bmatrix}$$

This matrix assumes a more interesting form when the measurements are made in one coordinate system and predictions in another. (As noted below, the vehicle lag coordinates are considered measured.)

The estimated error-covariance matrix E_e is interpreted as a weighted average of the prediction and measurement errors:

$$E_e = (1 - CH)E_p(1 - CH)^T + CE_m C^T \quad .$$

(This formula can be derived by looking at the variance of Z_e.) It is highly desirable to keep the estimated error-covariance matrix tied to the errors actually measured by the AMRAD radar. Accordingly, the measurement errors are approximated with

$$E_m \approx (Z_m - HZ_p)(Z_m - HZ_p)^T \quad .$$

That is, the variance in the coordinate measurements is set equal to the difference between the measured and predicted coordinates.

e. Nature of the C Matrix

The error-correction matrix C performs the following four basic operations on the vehicle state vector.

Correct Predicted Position--The estimated vehicle position is equal to the predicted position plus a small correction. The correction, in turn, is proportional to the difference between the measured and predicted positions. That is, the estimated position takes the basic form:

$$P_e = P_p + C_p(P_m - P_p)$$

where $0 < C_p < 1$. The magnitude of C_p is largely determined by the desired time constant for the tracking filter.

Correct Predicted Velocity--Upon reception of new AMRAD data, the C matrix calculates a new velocity estimate by adding a small correction to the predicted velocity. (As noted above, the predicted velocity equals the previously estimated velocity.) The correction, in turn, is proportional to the difference between the measured and predicted velocities:

$$\begin{aligned} V_e &= V_p + C_V(V_m - V_p) \\ &= V_p + C_V(P_m - (P_p - V_p) - V_p) \\ &= V_p + C_V (P_m - P_p) \quad . \end{aligned}$$

Note that V_p is the distance the vehicle travels in the time interval Δt. The constant C_V is chosen so that the filter remains stable. That is, C_V is chosen small enough--relative to C_p--to make the filter slightly less than critically damped.

Average Acceleration Lags--A Type-II filter makes no provision for vehicle acceleration. The predicted position may be expected to lag an accelerating vehicle and lead a decelerating one. The measure lag (or lead) is defined by

$$L_m = P_m - P_p \quad .$$

The error-correction matrix thus performs a weighted average of the predicted and measured acceleration lags:

$$L_e = L_p(1 - C_L) + C_L L_m$$

$$= L_p + C_L(L_m - L_p) \quad .$$

The constant C_L must be chosen large enough to obtain an up-to-date acceleration estimate and small enough to suppress fluctuations due to measurement error.

Average Measured SNRs--Like the acceleration lag coordinate, it is desirable to average the measured SNR over several radar returns to eliminate pulse-to-pulse fluctuations:

$$SNR_e = SNR_p + C_{SNR}(SNR_m - SNR_p) \quad .$$

The constant C_{SNR} must be small enough to smooth the measured data, but large enough to detect target fading when it occurs.

f. Form of the C Matrix

According to the above discussion, the C matrix takes the following sparse-matrix form:

$$\begin{bmatrix}
C_{PR} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & C_{PE} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & C_{PA} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
C_{VR} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & C_{VE} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & C_{VA} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & C_{LR} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & C_{LE} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & C_{LA} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & C_{SNR}
\end{bmatrix}$$

g. Variations of the C Matrix

The elements of the C matrix are varied according to (1) the amount of acceleration detected by the tracking filter and (2) the target's measured signal-to-noise ratio, as described below.

Acceleration Variations--During reentry the vehicle deceleration can become quite large. To follow such vehicles it is desirable to enlarge the elements of the C matrix when high decelerations are detected by the tracking filter:

$$C = \frac{A_o C_{LO} + A_p C_{HI}}{A_o + A_p} \quad .$$

This formula provides a continuous transition from the low-acceleration constants (C_{LO}) to the high-acceleration constants (C_{HI}), with the cross-over point determined by the constant A_o.

SNR Variations--Poor signal-to-noise ratios increase the error in the measured coordinates according to the usual radar-error formula:

$$\text{measurement error} = \text{antenna ambiguities} + \text{noise fluctuations} \quad .$$

The antenna ambiguities depend on the diameter of the radar aperture and the accuracy of its calibration. Noise fluctuations vary inversely as the square root of the SNR. To optimize the use of the radar measurements, the coordinate estimates should be made by forming a weighted average of the predicted and measured coordinates. The weighting, in turn, should depend on the relative errors in the predicted and measured coordinates. That is, the magnitude of the C matrix elements should vary directly with the ratio:

$$\frac{\text{prediction error}}{\text{prediction error + measurement error}} \quad .$$

Because the SNR can vary through several orders of magnitude as the target signal strengthens and fades, the above ratio looks like a step function when plotted against the possible values of the SNR. The step occurs when the noise fluctuations begin to dominate the antenna ambiguities in the radar-error formula quoted above. Accordingly, it is recommended that the elements of the C matrix depend on the SNR according to the following formula:

$$C = C_o \text{ step } (SNR - SNR_o) \quad .$$

Here step (x) equals 1 if $x > 0$, and equals 0 of $x < 0$.

C. TRANFT Flow Chart

To ease the programming effort and allow fast debugging, the program TRANFT was designed using modular, rather than in-line, code. A flow chart of TRANFT is shown in Figure B-1. A listing of TRANFT will be found at the end of this Appendix.

1. Initialize Filter (Begin, Start)

The tracking filter, vehicle state vector, and error-covariance matrices are initialized when TRANFT is first called. In particular, the subroutine BEGIN sets up the filter-constant matrix C, its identity-matrix complement CC, its transpose CT, and the identity-matrix complement of the transpose CCT. In addition, the rotation and translation constants needed to convert AMRAD measurements to HAPDAR coordinates are defined.

The tracking filter uses the first four sets of AMRAD data to initialize the vehicle state vectors and the covariance matrices. (The

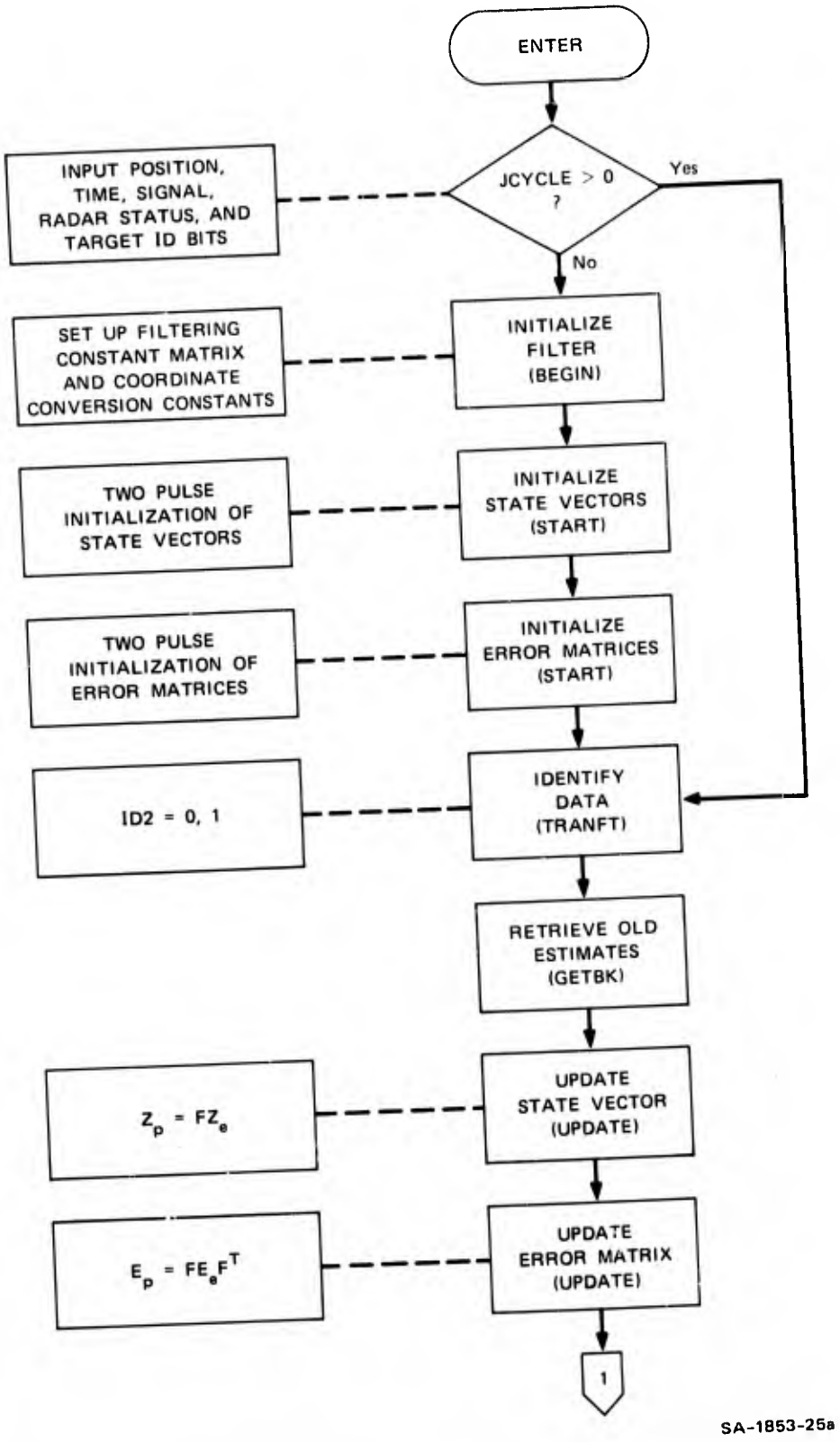


FIGURE B-1 BLOCK DIAGRAM OF TRANFT

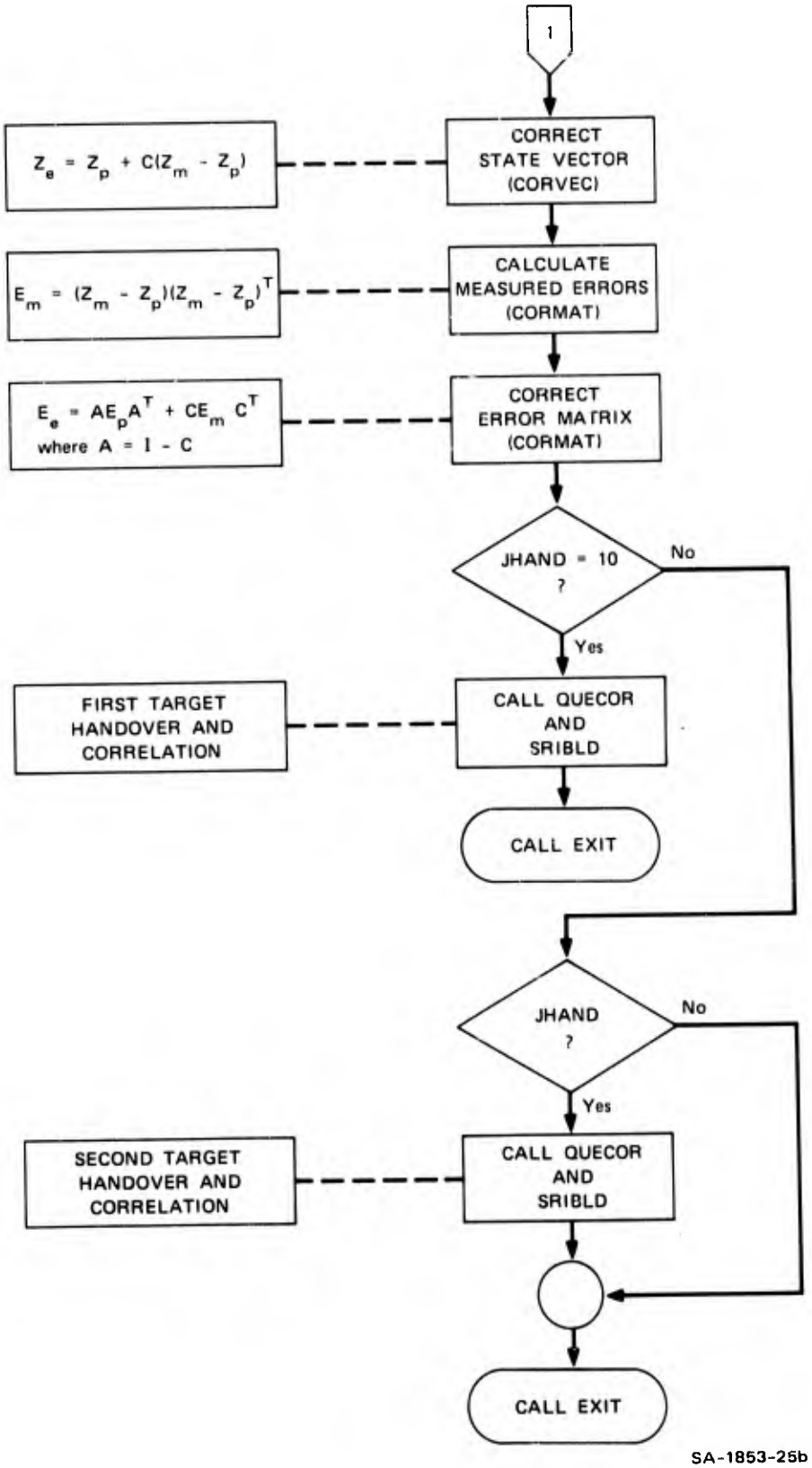


FIGURE B-1 BLOCK DIAGRAM OF TRANFT (Concluded)

first and third sets of data initialize the first vector and matrix, and the second and fourth sets initialize the second vector and matrix.) Two-pulse initialization is used to set up the position and velocity coordinates. The acceleration lag is set equal to 10 percent of the respective vehicle position coordinates, and the SNR is set equal to the last AMRAD reading.

The error covariance matrices are established by assuming an initial range error of 30 meters, an initial angle error of 4 milliradians, an initial velocity error of 20 times the respective position error, an initial lag error equal to the initial position error, and an initial SNR error equal to 10. Although somewhat arbitrary and artificial, these numbers worked out very satisfactorily in actual field tests.

2. Update Clock, Identify Data

Following filter, vector, and matrix initialization, the program TRANFT goes into a normal processing mode. As new data is received it is just used to update the program clock, as held in the word VTIME. Next, TRANFT determines whether the transmitted data applies to the first or second vehicle being tracked by AMRAD. Following the identification, the handover counters, I HAND or J HAND, are incremented and the identification and status bits stored.

3. Data Retrieval (GETBK)

The program TRANFT uses the identification word IDENT to retrieve the vehicle state vector and error-covariance matrix from common memory. Because vehicle velocities are stored in units of meters/second and filtering carried out in meters/pulse, time scaling is required for both the state vector and error matrices.

4. Predict Ahead (UPDATE)

Using the retrieved estimates, the program TRANFT next predicts ahead (actually updates) to the time of the newly received AMRAD data. The state vector prediction is accomplished by multiplying the estimated state vector times one F matrix. The error-covariance prediction is obtained by premultiplying by F and post-multiplying by the transpose of F.

5. Data Setup (SETUP)

The program TRANFT next calls SETUP to perform the required coordinate transformation between AMRAD and HAPDAR. (The coordinate transformation, which is performed in the subroutine COOR, is described in Ref. 2.) Following coordinate conversion, the measured data is set into the array DATA. During this operation the program compensates for any bias as scaling errors between the two radars. (These arise because of inaccurate surveys, misaligned clocks, and software bugs at HAPDAR.)

6. Vector Convection (CORVEC)

The program TRANFT next uses the measured data to correct the state-vector prediction made by UPDATE. The correction is made by adding part of the difference between the measured and predicted values to the predicted state vector. The filter constant matrix C determines the magnitude of the addition.

7. Matrix Correction (CORMAT)

The program TRANFT next uses the measured data to form a matrix of the measured errors. This matrix is formed by considering the differences between the measured and predicted vehicle coordinates. The subroutine CORMAT then corrects the predicted error-covariance matrix by adding a part of the difference between the predicted and measured errors

to the predicted error matrix. The magnitude of the addition is determined by the filter-constant matrix C.

8. Data Storage (PUTBK)

Following correction of the predicted state vector and error covariance matrix, the program TRANFT calls PUTBK to put the new estimates back in common memory. During this operation the data is scaled back from meters/pulse to meters/second, consistent with the BMDOS measurement units.

9. QUECOR, SRIBLD

Having completed its tasks, TRANFT next calls QUECOR and SRIBLD to carry out the vehicle correlation and track initialization, respectively. Following execution of these routines, TRANFT exits from the system.

A LISTING OF SUBROUTINE TRANFT

```
      PROGRAM MAIN(INPUT,OUTPUT)
C     THIS ROUTINE JUST CALLS THE SUBROUTINE TRANFT.
      DATA IPRT/0/
      CALL SECOND(T1)
      NN=54
      DO 10 I=1,NN
      CALL GETAM(IPRT)
      CALL TRANFT
10    CONTINUE
      CALL SECOND(T2)
      TE=T2-T1
      RE=TE/(NN-4)
      PRINT 75,T1,T2,TE,RE
75    FORMAT(1H ,4E15.7)
      END
      SUBROUTINE GETAM(IPRT)
C     THIS ROUTINE (NOT USED AT IBM 360/65) FILLS RSDS FOR CHECK-OUT
      DIMENSION IDATA(10)
      COMMON R11,R12,R13,R21,R22,R23,R31,R32,R33,ASCL,ANG,XAH,YAH,ZAH
      COMMON/RSDS/DUMMY1,VRANGE,VAZIMH,VELEV,VSNR,DUMMY2,VTIME,
     1IS1,IS2,ID1,ID2
      DOUBLE PRECISION TIME,VTIME,VEC1,TIME1,ECM1,VEC2,TIME2,ECM2
      DATA TWOPI/6.2831853070/
C     DEFINITION OF ELEMENTS OF VECTOR IDATA --
C     1 = AMRAD RANGE (LEAST COUNT = 1.8737055/1.25 METERS)
C     2 = AMRAD ELEVATION (LEAST COUNT = 1/16777216 OF A CIRCLE)
C     3 = AMRAD AZIMUTH (LEAST COUNT = 1/16777216 OF A CIRCLE)
C     4 = HAPDAR SIGNAL LEVEL (AGC SETTING IN DBSNR)
C     5 = RANGE TIME (LEAST COUNT = ONE MILLISECOND)
C     6 = DATA SOURCE (1 = AMRAD, 0 = RAS)
C     7 = TARGET (0= PRIMARY TARGET, 1 =  SECONDARY TARGET)
C     8 = TRACKING MODE (00 = MANUAL,01 = ACQUISITION, 11 = AUTOTRACK)
C     9 = UNUSED
C     10 = UNUSED
C
      READ 1,(IDATA(I),I=1,8)
1     FORMAT(10I8)
      IDATA(9)=1
C
      ASCL=1.8737055/1.25
      VRANGE=ASCL*IDATA(1)
      VAZIMH=TWOPI*IDATA(3)/16777216.
      VELEV=TWOPI*IDATA(2)/16777216.
      VSNR=IDATA(4)
      VTIME=IDATA(5)/2000.
      ID1=IDATA(6)
      ID2=IDATA(7)
      IS1=IDATA(8)
      IS2=IDATA(9)
C**************
C     DON SITE CHECK OUT
      IF(0) 111,111,112
111   VRANGE=16809.3829
      VAZIMH=2.058986-TWOPI/2.
      VELEV=6.279734
C**************
112   IF(IPRT) 20,20,1
10    PRINT 451,VRANGE,VAZIMH,VELEV,VSNR,VTIME,IS1,IS2,ID1,ID2
 451  FORMAT(1H ,4E12.5,D12.5,6I6)
20    RETURN
      END
      SUBROUTINE SRIHLD
C     DUMMY SUBROUTINE
```

```
      RETURN
      END
      SUBROUTINE QUECOR
C     DUMMY SUBROUTINE
      RETURN
      END
      SUBROUTINE TRANFT
C
C     THIS ROUTINE TRANFORMS MEASUREMENTS FROM AMRAD TO RADAR, AND THEN
C     FITS POSITON AND VELCITY WITH A TYPE-2 FILTER, ACCELERATION LAG
C     WITH A TYPE-1 FILTER, AND SNR WITH A TYPE-1 FILTER.
C
      DIMENSION STATE(10),ERROR(10,10),C(10,10),
     1  DATA(10),VEC1(10),ECM1(10,10),
     2VEC2(10),ECM2(10,10),CT(10,10),CC(10,10),CCT(10,10)
C
      COMMON R11,R12,R13,R21,R22,R23,R31,R32,R33,ASCL,ANG,XAH,YAH,ZAH
      COMMON/RSDS/DUMMY1,VRANGE,VAZIMH,VELEV,VSNR,DUMMY2,VTIME
     1IS1,IS2,ID1,ID2
      COMMON/AMTRK/VEC1,TIME1,IS11,IS12,ID11,ID12,ECM1,
     1VEC2,TIME2,IS21,IS22,ID21,ID22,ECM2,IDUM1,IDUM2
C
C     SET IPRT = 1 FOR PRINT OUT. SET IPRT = 0 FOR NO PRINT OUT.
      DATA ICYCLE,IPRT/-4,1/
C
      DOUBLE PRECISION TIME,VTIME,VEC1,TIME1,ECM1,VEC2,TIME2,ECM2
C
1     FORMAT(1H ,6I10)
2     FORMAT(1H ,4D12.5,4E12.5)
C
C     ICYCLE = NUMBER OF TIMES RSDS HAS BEEN READ
      ICYCLE=ICYCLE+1
      IF(ICYCLE) 3,3,5
3     IF(ICYCLE-2) 13,14,14
C
C     INITIALIZE PROGRAM
C     INITIALIZE FILTERING MATRICES.
C     TRANP AND H NO LONGER USED BY THIS PROGRAM.
13    CALL BEGIN(C,CT,CC,CCT,IPRT)
      CALL PUTAM(IDENT,STATE,C,IPRT)
      CALL PUTAM(IDENT,STATE,CT,IPRT)
      CALL PUTAM(IDENT,STATE,CC,IPRT)
      CALL PUTAM(IDENT,STATE,CCT,IPRT)
C
C     IHAND = COUNTER FOR FIRST VEHICLE
      IHAND=-10
C     JHAND = COUNTER FOR SECOND VEHICLE
      JHAND=-20
C
C     INITIALIZE STATE VECTORS AND ERROR COVARIANCE MATRICES.
14    CALL START(ICYCLE,IPRT)
C
C     TIME = PROGRAM TIME. VTIME = RSDS TIME.
      TIME=VTIME
      RETURN
C
C     FILTER INCOMING DATA.
C
C     IDENT = 1 FOR VEHICLE ONE, IDENT = 2 FOR VEHICLE TWO.
5     IDENT=ID2+1
      IF(IDENT-1) 6,6,10
C
C     FIRST TARGET DATA.
6     DELTA=VTIME-TIME1
      TIME=VTIME
      TIME1=TIME
```

```
      IHAND=IHAND+1
      IS11=IS1
      IS12=IS2
      ID11=ID1
      ID12=ID2
      GO TO 19
C
C     SECOND TARGET DATA.
10    DELTA=VTIME-TIME2
      TIME=VTIME
      TIME2=TIME
      JHAND=JHAND+1
      IS21=ID1
      IS22=ID2
      ID21=ID1
      ID22=ID2
C
C     GET OLD ESTIMATES FROM AMTRK FILE.
19    CALL GETBK(IDENT,STATE,ERROR)
      CALL PUTAM(IDENT,STATE,ERROR,IPRT)
C
C     UPDATE OLD ESTIMATES BY ONE PULSE TIME.
23    CALL UPDATE (STATE,ERROR,DELTA)
      CALL PUTAM(IDENT,STATE,ERROR,IPRT)
C
C     SET UP THE MESUREMENT VECTOR FOR FILTERING.
      CALL SETUP(STATE,DATA,IPRT)
      CALL PUTAM(IDENT,DATA ,ERROR,IPRT)
C
C     CORRECT THE UPDATED STATE ESTIMATES USING THE NEW MEASUREMENTS.
25    CALL CORVEC(STATE,C,DATA)
      CALL PUTAM(IDENT,STATE,ERROR,IPRT)
C
C     CORRECT THE UPDATED ERROR ESTIMATES USING THE NEW MEASUREMENTS.
      CALL CORMAT(STATE,DATA,ERROR,C,CC,CT,CCT)
      CALL PUTAM(IDENT,STATE,ERROR,IPRT)
C
C     PRINT OUT KEY NUMBERS
      IF(IPRT) 72,72,71
71    PRINT 1, ICYCLE,IHAND,JHAND,IDENT
      PRINT 1,IS1,IS2,ID1,ID2
      PRINT 1,IS11,IS12,ID11,ID12
      PRINT 1,IS21,IS22,ID21,ID22
      PRINT 2,VTIME,TIME,TIME1,TIME2,VRANGE,VAZIMH,VELEV,VSNR
C
C     PUT NEW STATE VECTOR AND ERROR MATRIX BACK INTO AMTRK FILE.
72    CALL PUTBK(IDENT,STATE,ERROR)
C
C     GENERATE OTDS FILE FOR FIRST VEHICLE.
      IF(IHAND-20) 92,91,92
91    IHAND=0
C     IDUM2=1
      CALL QUECOR
      CALL SRIBLD
      RETURN
C
C     GENERATE OTDS FILE FOR SECOND FILE.
92    IF(JHAND-20) 100,95,100
95    JHAND=0
C     IDUM2=2
      CALL QUECOR
      CALL SRIBLD
100   RETURN
      END
      SUBROUTINE BEGIN(C,CT,CC,CCT,IPRT)
C
```

```
C       ROUTINE TO INITIALIZE FILTERING MATRICES AND CALCULATION CONSTANTS.
C
        DIMENSION C(10,10),CT(10,10),CC(10,10),CCT(10,10)
        COMMON R11,R12,R13,R21,R22,R23,R31,R32,R33,ASCL,ANG,XAH,YAH,ZAH
C
C       READ IN ROTATION MATRIX
        R11=-6.5057487405E-01
        R12=7.5944017754E-01
        R13=1.7175573352E-03
        R21=-6.5637216849E-01
        R22=-5.6341633147E-01
        R23=5.[illegible]73460501E-01
        R31=3.[illegible]0511736E-01
        R32=3.2528857063E-01
        R33=8.6501990505E-01
C
C       SET VECTOR DISTANCE BETWEEN RADARS
        PI=3.1415926535
        RAD=2.*PI/360.
        SCL=12.*2.540005/100.
        RAH=55156.172*SCL
        EAH=-.11119*RAD
        AAH=220.64887*RAD
        SEAH=SIN(EAH)
        CEAH=COS(EAH)
        XAH=RAH*CEAH
        YAH=0.0
        ZAH=RAH*SEAH
        ANG=AAH-PI/2.
        ASCL=1.8737055/1.25
C
C       DEFINE FILTERING-CONSTANTS MATRICES.
C       C = FILTERING-CONSTANTS MATRIX
C       CT = TRANSPOSE OF OF C
C       CC = IDENTITY-MATRIX SCOMPLIMENTS OF C
C       CCT = TRANSPOSE OF CC
        DO 10 I=1,10
        DO 9  J=1,10
        C(I,J)=0.
9       CC(I,J)=0.0
10      CONTINUE
        C(1,1)=0.405
        C(2,2)=0.405
        C(3,3)=0.405
        C(4,1)=0.625
        C(5,2)=0.625
        C(6,3)=0.625
        C(7,7)=0.500
        C(8,8)=0.500
        C(9,9)=0.500
        C(10,10)=0.500
        DO 20 I=1,10
        DO 15 J=1,10
        IF(I-J) 13,14,13
13      CC(I,J)=-C(I,J)
        GO TO 15
14      CC(I,J)=1.0-C(I,J)
15      CONTINUE
20      CONTINUE
        DO 55 I=1,10
        DO 54 J=1,10
        CCT(I,J)=CC(J,I)
54      CT(I,J)=C(J,I)
55      CONTINUE
C
C       PRINT OUT TRANSFORMATION NUMBERS.
```

```
      IF(IPRT) 556,556,555
555   PRINT      1 ,R11,R12,R13,R21,R22,R23,R31,R32,R33
      PRINT 1,ASCL,ANG,XAH,YAH,ZAH
1     FORMAT(4E20.10)
556   RETURN
      END
      SUBROUTINE START(ICYCLE,IPRT)
C
C     SUBROUTINE  TO INITIALIZE STATE VECTORS AND ERROR MATRICES.
C
      DIMENSION VEC1(10),VEC2(10),ECM1(10,10),ECM2(10,10)
      COMMON/RSDS/DUMMY1,VRANGE,VAZIMH,VELEV,VSNR,DUMMY2,VTIME,
     1IS1,IS2,ID1,ID2
      COMMON/AMTRK/VEC1,TIME1,IS11,IS12,ID11,ID12,ECM1,
     1VEC2,TIME2,IS21,IS22,ID21,ID22,ECM2
C
      DOUBLE PRECISION TIME,VTIME,VEC1,TIME1,ECM1,VEC2,TIME2,ECM2
C
C     CONVERT AMRAD MEASUREMENTS TO HAPDAR COORDINATES
      CALL COOR(VRANGE,VAZIMH,VELEV,RBAR,ABAR,BBAR,IPRT)
C
C     INITIALIZE STATE VECTOR USING RSDS NUMBERS.
      IF(ICYCLE+2) 20,30,5
20    VEC1(1)=RBAR
      VEC1(2)=ABAR
      VEC1(3)=BBAR
      RETURN
30    VEC2(1)=RBAR
      VEC2(2)=ABAR
      VEC2(3)=BBAR
      RETURN
5     IF(ICYCLE+1) 10,10,15
10    VEC1(4)=-10.*(VEC1(1)-RBAR)
      VEC1(5)=-10.*(VEC1(2)-ABAR)
      VEC1(6)=-10.*(VEC1(3)-BBAR)
      RETURN
15    VEC2(4)=-10.*(VEC2(1)-RBAR)
      VEC2(5)=-10.*(VEC2(2)-ABAR)
      VEC2(6)=-10.*(VEC2(3)-BBAR)
C
C     INITIALIZE STATUS AND ID HITS.
      IS11=0
      IS12=1
      IS21=0
      IS22=1
      ID11=1
      ID12=0
      ID21=1
      ID22=1
C
C     INITIALIZE TIMES
      TIME1=VTIME
      TIME2=VTIME
C
      VEC1(7)=0.1*RBAR
      VEC2(7)=0.1*RBAR
      VEC1(8)=0.1*ABAR
      VEC2(8)=0.1*ABAR
      VEC1(9)=0.1*BBAR
      VEC2(9)=0.1*BBAR
      VEC1(10)=VSNR
      VEC2(10)=VSNR
C
C     INITIALIZE ERROR MATRICES USING STATE VECTOR NUMBERS.
      DO 55 I=1,10
      DO 50 J=1,10
```

```
      ECM1(I,J)=0.0
50    ECM2(I,J)=0.0
55    CONTINUE
      ECM1(1,1)=30.**2
      ECM2(1,1)=30.**2
      ECM1(2,2)=(4.0/1000.)**2
      ECM2(2,2)=(4.0/1000.)**2
      ECM1(3,3)=(4.0/1000.)**2
      ECM2(3,3)=(4.0/1000.)**2
      ECM1(4,4)=400.*ECM1(1,1)
      ECM2(4,4)=400.*ECM2(1,1)
      ECM1(5,5)=400.*ECM1(2,2)
      ECM2(5,5)=400.*ECM2(2,2)
      ECM1(6,6)=400.*ECM1(3,3)
      ECM2(6,6)=400.*ECM2(3,3)
      ECM1(7,7)=ECM1(1,1)
      ECM2(7,7)=ECM2(1,1)
      ECM1(8,8)=ECM1(2,2)
      ECM2(8,8)=ECM2(2,2)
      ECM1(9,9)=ECM1(3,3)
      ECM2(9,9)=ECM2(3,3)
      ECM1(10,10)=10.**2
      ECM2(10,10)=10.**2
100   CONTINUE
      RETURN
      END
      SUBROUTINE GETBK(IDENT,STATE,ERROR)
C
C     ROUTINE TO GET INFORMATION BACK FROM AMTRK
C
      DIMENSION STATE(10),ERROR(10,10),
     1VEC1(10),ECM1(10,10),VEC2(10),ECM2(10,10)
      COMMON/AMTRK/VEC1,TIME1,IS11,IS12,ID11,ID12,ECM1,
     1VEC2,TIME2,IS21,IS22,ID21,ID22,ECM2,IDUM1,IDUM2
C
      DOUBLE PRECISION TIME,VTIME,VEC1,TIME1,ECM1,VEC2,TIME2,ECM2
C
      ICODE=0
      IF(IDENT-1) 100,100,200
100   CALL SCALE(STATE,VEC1,ERROR,ECM1,ICODE)
      RETURN
200   CALL SCALE(STATE,VEC2,ERROR,ECM2,ICODE)
      RETURN
      END
      SUBROUTINE SETUP (STATE,DATA,IPRT)
C
C     ROUTINE TO SET UP VECTOR DATA WITH AMRAD MEASUREMENTS.
C
      DIMENSION STATE(10),DATA(10)
      COMMON/ RSDS/DUMMY1,VRANGE,VAZIMH,VELEV,VSNR,DUMMY2,VTIME,
     1IS1,IS2,ID1,ID2
      DOUBLE PRECISION TIME,VTIME,VEC1,TIME1,ECM1,VEC2,TIME2,ECM2
C
C     DEFINE CALIBRATION NUMBERS FROM PROGRAM BIASES.
      DATA AR1,AR2,AR3/0.0,1.0,0.0/
      DATA AU1,AU2,AU3/0.0,1.0,0.0/
      DATA AV1,AV2,AV3/0.0,1.0,0.0/
      DATA ASNR/0.0/
C
C     PERFORM COORDINATE TRANSFORMATION FROM AMRAD TO HAPDAR.
      CALL COOR(VRANGE,VAZIMH,VELEV,RRR,UUU,VVV,IPRT)
C
C     SET UP VECTOR DATA.
C     1 = RANGE, 2 = U, 3 = V, 4 = RDOT, 5 = UDOT, 6 = VDOT,
C     7 = RANGE LAG, 8 = U-LAG = 9 = V-LAG , 10 = SNR
      DATA(1)=AR1+AR2*RRR+AR3*VVV
```

```
      DATA(2)=AU1+AU2*UUU+AU3*VVV
      DATA(3)=AV1+AV2*VVV+AV3*UUU
      DATA(4)=STATE(4)
      DATA(5)=STATE(5)
      DATA(6)=STATE(6)
      DATA(7)=DATA(1)-STATE(1)
      DATA(8)=DATA(2)-STATE(2)
      DATA(9)=DATA(3)-STATE(3)
      DATA(10)=VSNR+ASNR
      RETURN
      END
      SUBROUTINE COOR(VRANGE,VAZIMH,VELEV,RRR,UUU,VVV,IPRT)
C
C     ROUTINE TO PERFORM COORDINATE TRANSFORMATION FROM AMRAD TO HAPDAR.
C
      COMMON R11,R12,R13,R21,R22,R23,R31,R32,R33,ASCL,ANG,XAH,YAH,ZAH
      IF(IPRT) 20,20,10
10    PRINT 1,VRANGE,VAZIMH,VELEV
20    RAT=VRANGE
      AAT=VAZIMH
      EAT=VELEV
      AAT=AAT-ANG
      CAAT=COS(AAT)
      SAAT=SIN(AAT)
      CEAT=COS(EAT)
      SEAT=SIN(EAT)
      XAT=RAT*CEAT*SAAT
      YAT=RAT*CEAT*CAAT
      ZAT=RAT*SEAT
      XSAVE=XAT-XAH
      YSAVE=YAT-YAH
      ZSAVE=ZAT-ZAH
      XFT=R11*XSAVE+R12*YSAVE+R13*ZSAVE
      YFT=R21*XSAVE+R22*YSAVE+R23*ZSAVE
      ZFT=R31*XSAVE+R32*YSAVE+R33*ZSAVE
C
C     RRR  = HAPDAR RANGE COORDINATE
      RRR =SQRT(XFT*XFT+YFT*YFT+ZFT*ZFT)
C
C     UUU  = HADAR U COORDINATE
      UUU =-ZFT/RRR
C
C     VVV  = HADAR V COORDINATE
      VVV =+XFT/RRR
      IF(IPRT) 40,40,30
30    PRINT 1,RRR,UUU,VVV
1     FORMAT(1H ,4E12.5)
40    RETURN
      END
      SUBROUTINE UPDATE (STATE,ERROR,DELTA)
C
C     ROUTINE TO UPDATE STATE VECTOR AND ERROR MATRIX BY ONE PULSE PERIOD.
C     NOMINAL PULSE PERIOD EQUALS 0.1 SECONDS
C
      DIMENSION STATE(10),ERROR(10,10)
C
C     UPDATE STATE VECTOR USING DELTA
      RATIO=DELTA/0.1
      STATE(1)=STATE(1)+STATE(4)*RATIO
      STATE(2)=STATE(2)+STATE(5)*RATIO
      STATE(3)=STATE(3)+STATE(6)*RATIO
C
C     UPDATE ERROR MATRIX USING DELTA
      DO 10 I=1,3
      DO 9 J=1,6
9     ERROR(I,J)=ERROR(I,J)+ERROR(I+3,J)*(RATIO*RATIO)
```

```
10      CONTINUE
        DO 20 I=1,6
        DO 19 J=1,3
19      ERROR(I,J)=ERROR(I,J)+ERROR(I,J+3)*(RATIO*RATIO)
20      CONTINUE
        RETURN
        END
        SUBROUTINE CORVEC(STATE,C,DATA)
C
C       ROUTINE TO CORRECT UPDATED STATE VECTOR WITH MEASURED DATA.
C       STATE(NEW) = STATE(OLD) + C*(STATE(MEASURED)-STATE(OLD))
C                  = (1 - C)*STATE(OLD) + C*STATE(MEASURED)
C                  = CC*STATE(OLD) + C*STATE(MEASURSED)
C
        DIMENSION STATE(10),C(10,10),DATA(10)
        STATE(4)=STATE(4)+C(4,1)*(DATA(1)-STATE(1))
        STATE(5)=STATE(5)+C(5,2)*(DATA(2)-STATE(2))
        STATE(6)=STATE(6)+C(6,3)*(DATA(3)-STATE(3))
        STATE(1)=STATE(1)+C(1,1)*(DATA(1)-STATE(1))
        STATE(2)=STATE(2)+C(2,2)*(DATA(2)-STATE(2))
        STATE(3)=STATE(3)+C(3,3)*(DATA(3)-STATE(3))
        STATE(7)=STATE(7)+C(7,7)*(DATA(7)-STATE(7))
        STATE(8)=STATE(8)+C(8,8)*(DATA(8)-STATE(8))
        STATE(9)=STATE(9)+C(9,9)*(DATA(9)-STATE(9))
        STATE(10)=STATE(10)+C(10,10)*(DATA(10)-STATE(10))
        RETURN
        END
        SUBROUTINE  CORMAT(STATE,DATA,ERROR,C,CC,CT,CCT)
C
C       ROUTINE TO CORRECT UPDATED ERROR COVARIANCE MATRIX WITH MEASURED DATA.
C
        DIMENSION STATE(10),ERROR(10,10),C(10,10),CC(10,10),
       1DATA(10),DIFF(10),RMAT(10,10),SMAT(10,10),TMAT(10,10),TVEC(10)
       2,CCT(10,10),CT(10,10),UMAT(10,10),VMAT(10,10)
C
C       FORM MEASURED ERROR COVARIANCE MATRIX AND STORE IN TMAT.
        DO 10 I=1,10
10      DIFF(I)=DATA(I)-STATE(I)
        DO 20 I=1,10
        DO 15 J=1,10
        RMAT(I,J)=0.0
        SMAT(I,J)=0.0
        TMAT(I,J)=DIFF(I)*DIFF(J)
        UMAT(I,J)=0.0
15      VMAT(I,J)=0.0
20      CONTINUE
C
C       DETERMINE NEW ESTIMATE FOR ERROR COVARIANCE MATRIX.
        IF(1) 200,200,300
200     CALL MULMAT(SMAT,C,TMAT)
        CALL MULMAT(RMAT,SMAT,CT)
        CALL MULMAT(VMAT,CC,ERROR)
        CALL MULMAT(UMAT,VMAT,CCT)
        GO TO 91
C
300     DO 25 I=1,6
        SMAT(1,I)=C(1,1)*TMAT(1,I)
        SMAT(2,I)=C(2,2)*TMAT(2,I)
        SMAT(3,I)=C(3,3)*TMAT(3,I)
        SMAT(4,I)=C(4,1)*TMAT(1,I)
        SMAT(5,I)=C(5,2)*TMAT(2,I)
25      SMAT(6,I)=C(6,3)*TMAT(3,I)
        DO 26 I=7,9
        SMAT(7,I)=C(7,7)*TMAT(7,I)
        SMAT(8,I)=C(8,8)*TMAT(8,I)
26      SMAT(9,I)=C(9,9)*TMAT(9,I)
```

```
      SMAT(10,10)=C(10,10)*TMAT(10,10)
C
      DO 27 I=1,6
      RMAT(I,1)=SMAT(I,1)*CT(1,1)
      RMAT(I,2)=SMAT(I,2)*CT(2,2)
      RMAT(I,3)=SMAT(I,3)*CT(3,3)
      RMAT(I,4)=SMAT(I,1)*CT(1,4)
      RMAT(I,5)=SMAT(I,2)*CT(2,5)
27    RMAT(I,6)=SMAT(I,3)*CT(3,6)
      DO 28 I=7,9
      RMAT(I,7)=SMAT(I,7)*CT(7,7)
      RMAT(I,8)=SMAT(I,8)*CT(8,8)
28    RMAT(I,9)=SMAT(I,9)*CT(9,9)
      RMAT(10,10)=SMAT(10,10)*CT(10,10)
C
      DO 29 I=1,6
      VMAT(1,I)=CC(1,1)*ERROR(1,I)
      VMAT(2,I)=CC(2,2)*ERROR(2,I)
      VMAT(3,I)=CC(3,3)*ERROR(3,I)
      VMAT(4,I)=CC(4,1)*ERROR(1,I)+CC(4,4)*ERROR(4,I)
      VMAT(5,I)=CC(5,2)*ERROR(2,I)+CC(5,5)*ERROR(5,I)
29    VMAT(6,I)=CC(6,3)*ERROR(3,I)+CC(6,6)*ERROR(6,I)
      DO 30 I=7,9
      VMAT(7,I)=CC(7,7)*ERROR(7,I)
      VMAT(8,I)=CC(8,8)*ERROR(8,I)
30    VMAT(9,I)=CC(9,9)*ERROR(9,I)
      VMAT(10,I)=CC(10,10)*ERROR(10,I)
C
      DO 31 I=1,6
      UMAT(I,1)=VMAT(I,1)*CCT(1,1)
      UMAT(I,2)=VMAT(I,2)*CCT(2,2)
      UMAT(I,3)=VMAT(I,3)*CCT(3,3)
      UMAT(I,4)=VMAT(I,1)*CCT(1,4)+VMAT(I,4)*CCT(4,4)
      UMAT(I,5)=VMAT(I,2)*CCT(2,5)+VMAT(I,5)*CCT(5,5)
31    UMAT(I,6)=VMAT(I,3)*CCT(3,6)+VMAT(I,6)*CCT(6,6)
      DO 32 I=7,9
      UMAT(I,7)=VMAT(I,7)*CCT(7,7)
      UMAT(I,8)=VMAT(I,8)*CCT(8,8)
32    UMAT(I,9)=VMAT(I,9)*CCT(9,9)
      UMAT(I,10)=VMAT(I,10)*CCT(10,10)
51    DO 60 I=1,10
      DO 55 J=1,10
55    ERROR(I,J)=RMAT(I,J)+UMAT(I,J)
60    CONTINUE
      RETURN
      END
      SUBROUTINE PUTAM(IDENT,STATE,ERROR,IPRT)
C
C     SUBROUTINE TO PRINT CALCULATED NUMBERS.
C
      DIMENSION STATE(10),ERROR(10,10),TEMP(10)
      IF(IPRT) 100,100,200
200   PRINT 1
      PRINT 1,  (STATE(I),I=1,10)
1     FORMAT(1H ,10E12.5)
      DO 10 I=1,10
10    TEMP(I)=SQRT(ABS(ERROR(I,I)))
      PRINT 1
      PRINT 1,(TEMP(I),I=1,10)
      PRINT 1
      PRINT 1,((ERROR(I,J),J=1,10),I=1,10)
100   RETURN
      END
      SUBROUTINE  PUTBK(IDENT,STATE,ERROR)
C
C     ROUTINE TO PUT CALCULATED DATA BACK INTO AMTRK.
```

```
C
      DIMENSION STATE(10),ERROR(10,10),
     1VEC1(10),ECM1(10,10),VEC2(10),ECM2(10,10)
      COMMON/AMTRK/VEC1,TIME1,IS11,IS12,ID11,ID12,ECM1,
     1VEC2,TIME2,IS21,IS22,ID21,ID22,ECM2
C
      DOUBLE PRECISION TIME,VTIME,VEC1,TIME1,ECM1,VEC2,TIME2,ECM2
C
      ICODE=1
      IF(IDENT-1) 100,100,200
100   CALL SCALE(STATE,VEC1,ERROR,ECM1,ICODE)
      RETURN
200   CALL SCALE(STATE,VEC2,ERROR,ECM2,ICODE)
      RETURN
      END
      SUBROUTINE SCALE(STATE,VEC1,ERROR,ECM1,ICODE)
C
C     ROUTINE TO GET/PUT AMRAD DATA, WITH APPROPRIATE SCALING.
C
      DIMENSION STATE(10),VEC1(10),ERROR(10,10),ECM1(10,10)
      DOUBLE PRECISION TIME,VTIME,VEC1,TIME1,ECM1,VEC2,TIME2,ECM2
      IF(ICODE)1,1,100
C
C     SCALE VELOCITY FROM METERS/SEC TO METERS/PULSE.
1     DEL=0.1
      DO 5 I=1,10
      DO 4 J=1,10
4     ERROR(I,J)=ECM1(I,J)
5     STATE(I)=VEC1(I)
      DO 30 I=1,3
      DO 29 J=4,6
29    ERROR(I,J)=ERROR(I,J)*DEL
30    CONTINUE
      DO 28 I=4,6
      STATE(I)=STATE(I)*DEL
      DO 27 J=1,3
27    ERROR(I,J)=ERROR(I,J)*DEL
      DO 26 J=4,6
26    ERROR(I,J)=ERROR(I,J)*(DEL*DEL)
28    CONTINUE
      RETURN
C
C     SCALE VELOCITY FROM METERS/PULSE TO METERS/SEC.
100   DEL=10.
      DO 115 I=1,10
      DO 114 J=1,10
114   ECM1(I,J)=ERROR(I,J)
115   VEC1(I)=STATE(I)
      DO130 I=1,3
      DO129 J=4,6
129   ECM1 (I,J)=ECM1(I,J)*DEL
130   CONTINUE
      DO128 I=4,6
      VEC1 (I)=VEC1(I)*DEL
      DO127 J=1,3
127   ECM1 (I,J)=ECM1(I,J)*DEL
      DO126 J=4,6
126   ECM1 (I,J)=ECM1(I,J)*(DEL*DEL)
128   CONTINUE
      RETURN
      END
      SUBROUTINE MULMAT(A,B,C)
C
C     MATRIX MULTIPLICATION ROUTINE
C     NON-SYMMETRIC VERSION
C
```

```
      DIMENSION A(10,10),B(10,10),C(10,10)
      DO 2 I=1,10
      DO 1 J=1,10
1     A(I,J)=0.0
2     CONTINUE
      DO 10 I=1,6
      DO 9  J=1,6
      DO 8  K=1,6
      SAVE1=B(I,K)
      SAVE2=C(K,J)
      IF(SAVE1) 6,8,6
6     IF(SAVE2) 7,8,7
7     A(I,J)=A(I,J)+SAVE1*SAVE2
8     CONTINUE
9     CONTINUE
10    CONTINUE
      DO 20 I=7,9
      DO 19 J=7,9
      DO 18 K=7,9
18    A(I,J)=A(I,J)+B(I,K)*C(K,J)
19    CONTINUE
20    CONTINUE
      A(10,10)=B(10,10)*C(10,10)
      RETURN
      END
```

Appendix C

CORRELATION PROGRAM QUECOR

Appendix C

CORRELATION PROGRAM QUECOR

The field test version of the correlation algorithm is called QUECOR. The purpose of this appendix is to briefly discuss the theoretical basis for the correlation algorithm, describe the logical structure and operation of QUECOR, and present a listing of QUECOR. A more detailed discussion of the theory and operation of the correlation algorithm was presented in the final report of the Radar Netting Phase II effort.[1] Also, a description of QUECOR and its interfaces with other portions of the software system is contained in Ref. 8.*

A. A Summary of the Theoretical Basis of the Correlation Algorithm

The purpose of the correlation algorithm is to determine on request whether an object (or objects) being tracked by one radar is also being tracked by a second radar. It does this by comparing a given track from one radar (assumed to be a radar from which we wish to hand over the track) with the track file of a second radar.

The correlation algorithm is based on computing a quadratic function, Q, of the difference between the estimated position vector produced by one tracking filter and one produced by another tracking filter, and the associated covariance matrices for these estimates. This quadratic function, Q (hereafter called the correlation parameter), is then compared to a threshold to test the hypothesis that the two tracks are of

---

*In Ref. 8, QUECOR is referred to as QUECORR.

the same object. In practice, the correlation parameter should have a pair of subscripts to identify the pair of tracks for which it is computed (i.e., Q_{ij} is correlation parameter for the pair, track i and track ipts will be suppressed in this and following sections d to avoid ambiguity.

Let $\underset{\sim}{\hat{x}}^i$ be the estimated state vector obtained from track i, and C_i be the corresponding covariance matrix. Given two state vectors obtained from two tracks, let $\Delta\underset{\sim}{\hat{x}} = \underset{\sim}{\hat{x}}^i - \underset{\sim}{\hat{x}}^j$ and $C = C_i + C_j$. Then Q for that pair of tracks is defined as

$$Q = \Delta\underset{\sim}{\hat{x}}^T \, C^{-1} \, \Delta\underset{\sim}{\hat{x}} \quad . \tag{2}$$

Note that $\Delta\underset{\sim}{\hat{x}}^T$ is a (1 × 3) matrix, C^{-1} is a (3 × 3) matrix, and $\Delta\underset{\sim}{\hat{x}}$ is a (3 × 1) matrix. Thus, the product indicated by the right hand side of Eq. (2) is a (1 × 1) matrix, or a scalar. A nonmatrix form of Eq. (2) can be written if we define

$$\Delta\underset{\sim}{\hat{x}}^T = [\Delta\hat{x}_1, \; \Delta\hat{x}_2, \; \Delta\hat{x}_3]$$

and

$$C^{-1} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad .$$

Then, we have

$$Q = \sum_{\substack{k=1 \\ m=1}}^{3} a_{km} \, \Delta\hat{x}_k \, \Delta\hat{x}_m \quad . \tag{3}$$

The parameter Q can be thought of as the sum of the squares of the components of the position difference vector normalized by a measure of the expected dispersion in the corresponding component direction.

Since the components of the state vector $\underset{\sim}{\hat{x}}^i$ are random variables which are assumed to be normally distributed with covariance matrix C_i, Q also is a random variable whose probability density function can be derived. In the case where the mean value of $\Delta\underset{\sim}{\hat{x}}$ is the zero vector, the parameter Q has a chi-square distribution. This corresponds to the case where the two tracks being compared are correlated or correspond to the same object, and no bias errors are present between the two tracking radars. Due to the normalizing feature in the parameter Q, and the fact that in general Q is computed from an n-long vector, the mean value of Q turns out to be equal to n. The n is called the degree of freedom, and in our case n = 3, so that the mean value of Q is 3.

The chi-square density function is given by

$$f_{x^2}(n,Q) = \frac{Q^{n/2-1} \exp\left[-\frac{1}{2}Q\right]}{\Gamma\left(\frac{n}{2}\right) 2^{n/2}} \quad , \tag{4}$$

where $Q \geq 0$, and $\Gamma(n/2)$ is the gamma function, which for integer arguments reduces to the factorial function [i.e., if n is even then $\Gamma(n/2) = (n/2 - 1)!$]. For our case, n = 3, and $\Gamma(3/2) = \sqrt{\pi/2}$, and we obtain

$$f_{x^2}(3,Q) = \frac{Q^{1/2} \exp\left[-\frac{1}{2}Q\right]}{\sqrt{2\pi}} \quad . \tag{5}$$

The function is plotted in Figure C-1.

When the objects are uncorrelated, the quadratic function Q will be a sample from a noncentral chi-square distribution, which has an additional parameter called the noncentrality parameter, λ. The noncentrality parameter is a function of the separation vector between the pair of uncorrelated objects. In particular, it is the value of the quadratic function evaluated at the true separation vector. If the true separation vector is given by $\underset{\sim}{\Delta z}$, the λ is given by

$$\lambda = \underset{\sim}{\Delta z}^T \, C^{-1} \underset{\sim}{\Delta z} \tag{6}$$

and the noncentral chi-square density function is given by

$$f_{ncx^2}(n,\lambda,Q) = \sum_{r=0}^{\infty} \frac{1}{r!} \left(\frac{\lambda}{2}\right)^r \frac{Q^{n/2+r-1} \exp\left[-\frac{1}{2}(Q+\lambda)\right]}{\Gamma\left(\frac{n}{2}+r\right) 2^{n/2+r}} \quad . \tag{7}$$

For our case, where n = 3, we obtain

$$f_{ncx^2}(3,\lambda,Q) = \sum_{r=0}^{\infty} \frac{1}{r!} \left(\frac{\lambda}{2}\right)^r \frac{Q^{1/2+r} \exp\left[-\frac{1}{2}(Q+\lambda)\right]}{\Gamma\left(\frac{3}{2}+r\right) 2^{3/2+r}} \quad . \tag{8}$$

The form of this distribution in relation to the chi-square is shown in Figure C-2. Note that when $\lambda = 0$, Eqs. (7) and (8) reduce to the chi-square density functions. Thus, the chi-square distribution can be viewed as belonging to the more general family of noncentral chi-square distributions.

To summarize, when a value of Q is computed for a particular pair of tracks, it will either be a sample from a chi-square or noncentral chi-square, depending on whether the two tracks are correlated or not. The correlation algorithm applies a threshold hypothesis-testing approach

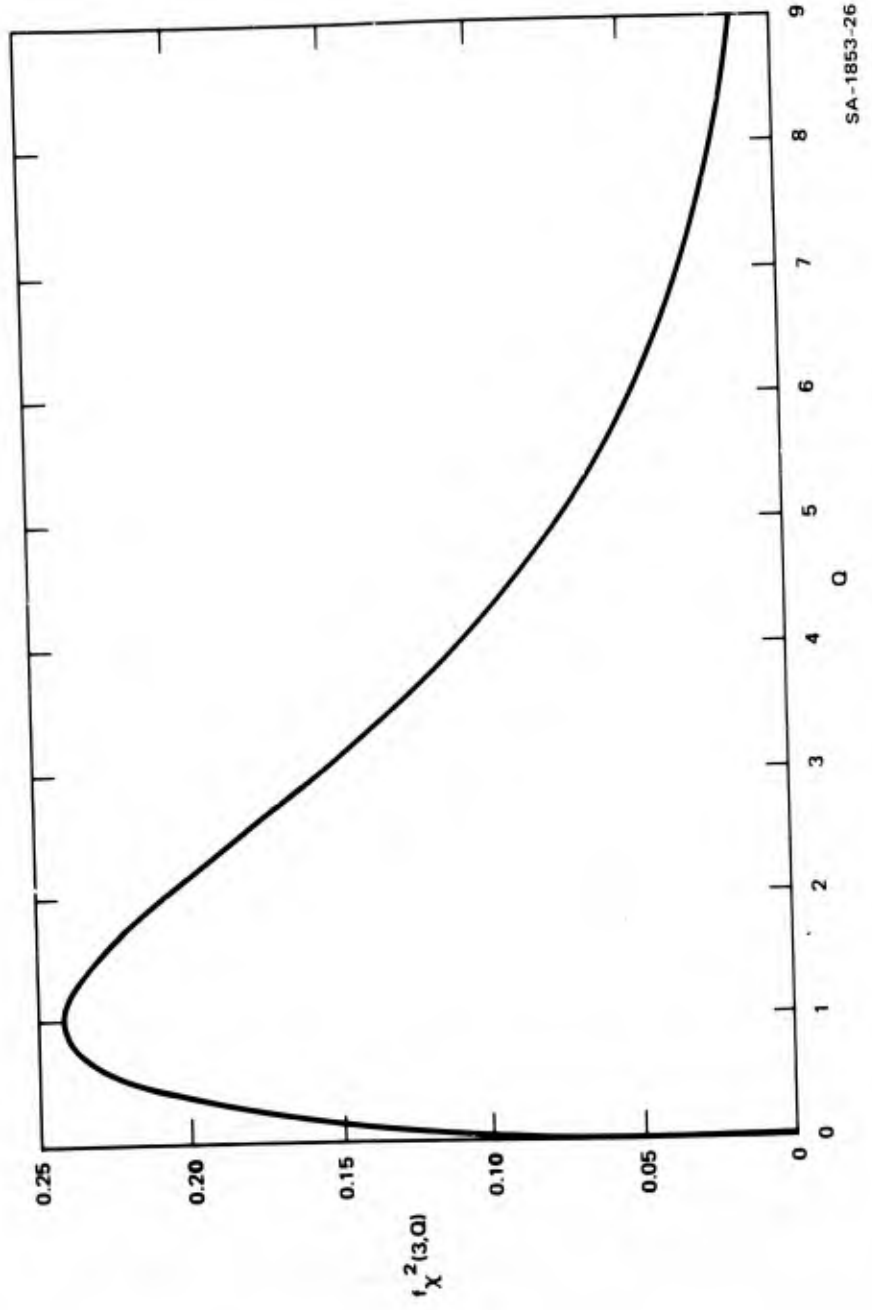


FIGURE C-1 CHI-SQUARE DENSITY FUNCTION

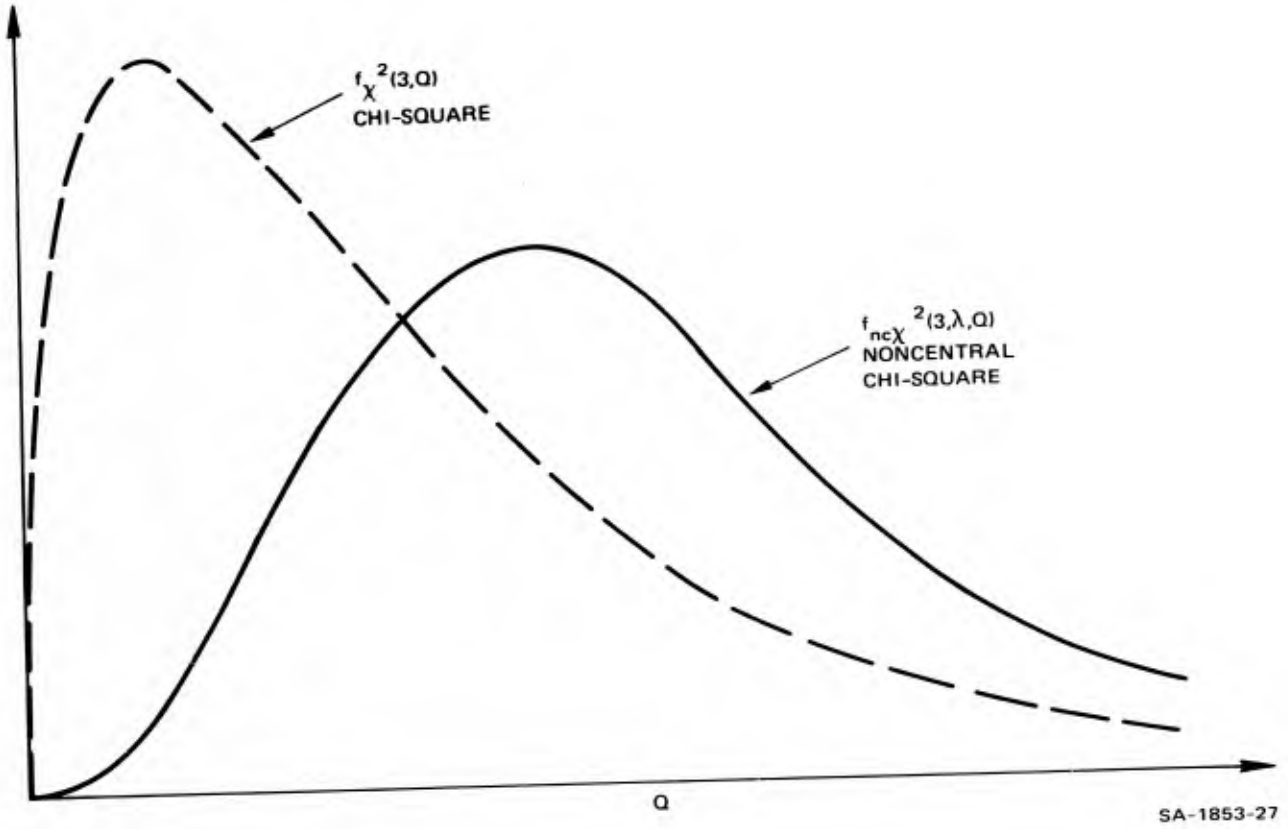


FIGURE C-2 NONCENTRAL CHI-SQUARE DENSITY FUNCTION

to infer which is the case. We test the hypothesis that the tracks are correlated by comparing the computed value of Q with a threshold value Q_t, and accept the hypothesis that the tracks are correlated if Q is less than Q_t. At the same time, we test the alternate hypothesis that the tracks are uncorrelated in the same manner but accept the alternate hypothesis only if Q is greater than Q_t. The problem then is to select a Q_t such that the probability of erroneously calling a correlated pair uncorrelated (a Type I error) and the probability of calling an uncorrelated pair correlated (a Type II error) are both acceptably low.

Obviously, as the true separation of objects in a multiobject situation becomes smaller and smaller, it will become impossible to select a Q_t that meets the above criterion. We must therefore postulate some minimum separation of objects in the tactical situation. As the separation between objects increases, the hump in the noncentral chi-square density moves farther to the right in Figure C-2 (i.e., the expected values of Q become larger). This fact, that the expected values of Q for the chi-square and the noncentral chi-square move farther and farther apart as the true separation of objects increases, motivates the threshold decision rule discussed above.

To determine a value of Q_t we can first compute the probability that $Q > Q_t$ for a sample from a chi-square distribution as a function of Q_t. Similarly, assuming some lower bound on the separation and its implications on a lower bound for λ, we can compute the probability that $Q < Q_t$ for a sample from a noncentral chi-square distribution as a function of Q_t. If these results indicate that there exists an interval of values of Q_t such that both probabilities are lower than some selected low probability, then any value of Q_t from this interval can be selected to meet the desired criterion.

In Ref. 2 a reasonable lower bound on λ of about 50 was derived, based on an assumed lower bound on the object separation of about ten times a radar's single hit measurement error standard deviation.

For this value of λ, the probability that a Q from a noncentral chi-square distribution would be less than 24.2 is 0.01. On the other hand, the probability that the Q for a correlated pair of tracks would be greater than about 11.35 is also 0.01. Thus, any value of Q_t between 11.35 and 24.2 would ensure that the probability of making a correlation error of either type would be less than or equal to 0.01. (Tables used for the calculation of these values can be found in Ref. 9 for the chi-square distribution, and in Ref. 10 for the noncentral chi-square distribution.)

It should be noted in addition to the fact that the case discussed above assumed a minimum separation, the estimation accuracy increases as more measurements are incorporated so that the expected value of Q for uncorrelated objects should increase as time progresses, while the expected value of Q for correlated objects remains stationary.

In the implementation of any type of threshold testing, it is useful to obtain bounds on the value of the tested parameter. This is true since these bounds may be much simpler to compute and may obviate the need for computing the actual values of the tested parameter in most cases. Thus, a net saving in computation time may be achieved.

Lower and upper bounds can be established on Q using the following inequality

$$\frac{\Delta\hat{\underset{\sim}{x}}^T \Delta\hat{\underset{\sim}{x}}}{U} \leq Q \leq \frac{\Delta\hat{\underset{\sim}{x}}^T \Delta\hat{\underset{\sim}{x}}}{L} \tag{9}$$

where U is an upper bound on the largest eigenvalue of the covariance C, and L is a lower bound on the smallest eigenvalue of C. Given C we can obtain U from

$$U = \max_{k=1,3} \left\{ c_{kk} + \sum_{\substack{m=1 \\ m \neq k}}^{3} |c_{km}| \right\} \tag{10}$$

where c_{km} = (k,m) element of C. Similarly we can obtain

$$L = \min_{k=1,3} \left\{ c_{kk} - \sum_{\substack{m=1 \\ m \neq k}}^{3} |c_{km}| \right\} . \tag{11}$$

These bounds on Q are simpler to compute than Q itself primarily due to the fact that they do not require the computation of the inverse of C and the pre- and post-multiplication of C^{-1} by $\Delta\underset{\sim}{\hat{x}}$.

The correlation algorithm theory discussed thus far has been developed under the assumption that there are no significant uncalibrated biases between the two radars employed in the handover process. With a careful periodic calibration of the two radars, it may be feasible to reduce the uncalibrated bias errors to a small part of the expected random errors. However, it is expected that significant uncalibrated or residual bias errors may remain. As some information on the statistics of these bias errors is available, a method is desired to either adjust the decision threshold or to adjust the computed Q values to account for these bias effects.

The first approach to adjusting the threshold has been investigated and reported in Refs. 1 and 2. An exact analysis was not feasible, but an analysis based on conservative simplifying assumptions was carried

out that resulted in a rationale for adjusting the decision threshold. This method entails obtaining the eigenvalues of C before computing Q, and obtaining the ratio of the minimum eigenvalues of C to the largest expected variance of the uncalibrated bias errors. This ratio can then be used for a table look-up of the proper Q threshold.

This method has the disadvantage of requiring the selection of a new decision threshold each time Q is calculated for each track pair being processed.

The second approach is to adjust the value of the computed Q. This can be implemented by adjusting the covariance C used in computing Q. The basic idea is to consider $\Delta\underset{\sim}{\hat{x}}$ as the sum of an unbiased difference vector and some bias vector. The covariance of the unbiased difference vector is given by C, and the covariance of the bias vector is given by B. B is assumed to be known. If the unbiased difference vector is independent of the bias vector, and the bias is assumed to be a sample from a zero mean normal with covariance B, then the covariance of $\Delta\underset{\sim}{\hat{x}}$ is equal to C + B. We can then compute Q as

$$Q = \Delta\underset{\sim}{\hat{x}}^{T} (C + B)^{-1} \Delta\underset{\sim}{\hat{x}} \quad . \tag{12}$$

Thus, when two tracks are correlated, the components of the difference vector $\Delta\underset{\sim}{\hat{x}}$ can be viewed as samples from a zero mean multivariate normal distribution with covariance C + B. Q would then be a sample from a chi-square distribution.

The effect of adjusting Q in such a manner is to decrease the computed value of Q to compensate for the expected increase in Q caused by any residual bias. This implies that a Q threshold can be selected before a mission or engagement, based on an analysis of the probability of Type I or Type II errors as previously outlined. However, any estimate of a

lower bound on the noncentrality parameter, λ, should be properly adjusted to account for the increased covariance when B is added to C. Note that the prior expected value of Q for correlated objects will be 3, but the posterior expected value after several Q computations for correlated objects will approach some other value depending on the particular value of the bias errors for a given period of operation of the system. This occurs because the bias errors are assumed to be constant during any typical continuous period of operation.

QUECOR has been implemented such that the covariance can be corrected as outlined above; however, the elements of B are currently set equal to zero pending handover calibration of the two radars and acquisition of statistical data on residual bias errors.

B. QUECOR Logic and Operation Description

QUECOR is a real time, field test implementation of the correlation algorithm whose theoretical basis has just been discussed. QUECOR is a routine that is executed on request whenever a handover of a track from one radar (AMRAD) to another (HAPDAR) is desired. For the field test demonstration, the system was designed to request handovers once every two seconds from each of two AMRAD trackers. These requests were scheduled by TRANFT, a program described in Appendix B. The handovers of the two tracks alternated at intervals of one second.

TRANFT initiates the handover process by calling QUECOR. QUECOR operates "open loop," so that the operation of all other functions is not affected by the output of QUECOR. TRANFT continues the handover process by calling SRIBLD which, among other functions, sets up the request for the first handover track pulse from HAPDAR. The other functions of SRIBLD include the initial building of the two dedicated track instances in the HAPDAR track file and the reinitialization of the dedicated tracks with data from the AMRAD tracks whenever a handover is scheduled.

A tactical version of QUECOR would also determine whether or not to continue the handover process, based on the results of the correlation logic, and would insure that a function like SRIBLD is called only when necessary.

The inputs required by QUECOR are contained in two data files, AMTRK and OTDS. AMTRK contains the data for the two AMRAD tracks, and OTDS is the track file for HAPDAR. OTDS contains the track data on the two dedicated tracks supporting the handover experiment and the track data on any other HAPDAR independently acquired track.

A flow chart for QUECOR is contained in Figure C-3, and a listing of the program appears at the end of this appendix. When QUECOR is called, various parameters are initialized and data from one of the track data sets in AMTRK is loaded into a buffer. The initializations basically consist of setting several threshold values and several covariance correction terms to correct for residual bias effects. The AMRAD track data loaded into the buffer correspond to that track most recently updated. In practice this results in the alternate selection of each of the two AMRAD tracks. Also, at this time QUECOR records the AMTRK file.

The data loaded into the buffer consist of the AMRAD track state, covariance, and time of validity. After the data are loaded, the covariance is corrected for bias effects by adding in the correction terms.

QUECOR next loads the track data from OTDS into a buffer. These data consist of the state, the covariance, the time of validity, the number of hits and misses, the off-boresight direction cosine of the estimated position, and a track identification number. The state vector is obtained in both an (R, u, v) coordinate system and a radar face (X, Y, Z) cartesian coordinate system. If the track is a polynomial filter, the covariance is in an (R, u, v) coordinate system; if the track is a Kalman filter, the covariance is in the (X, Y, Z) system.

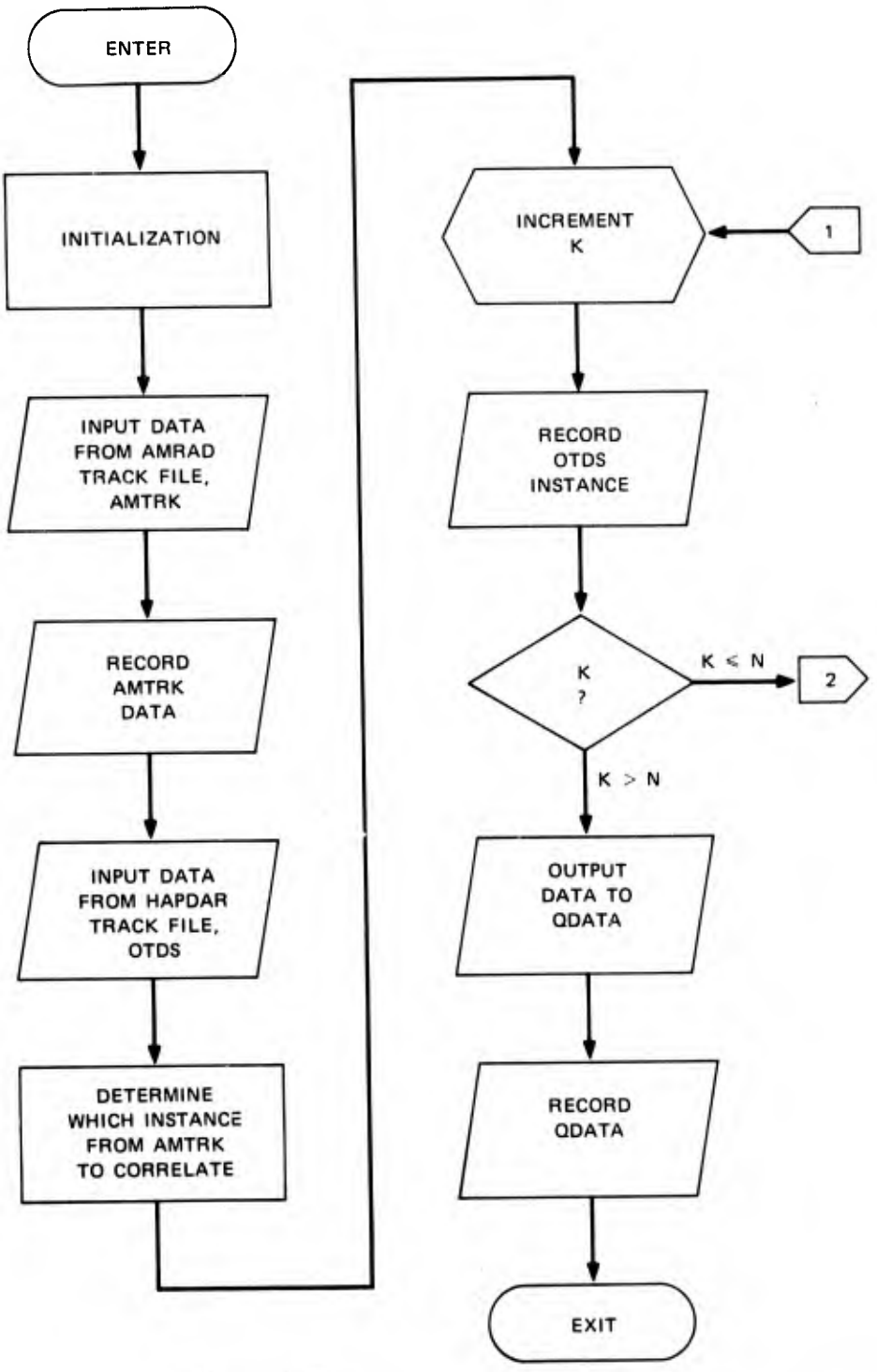


FIGURE C-3 QUECOR FLOW CHART

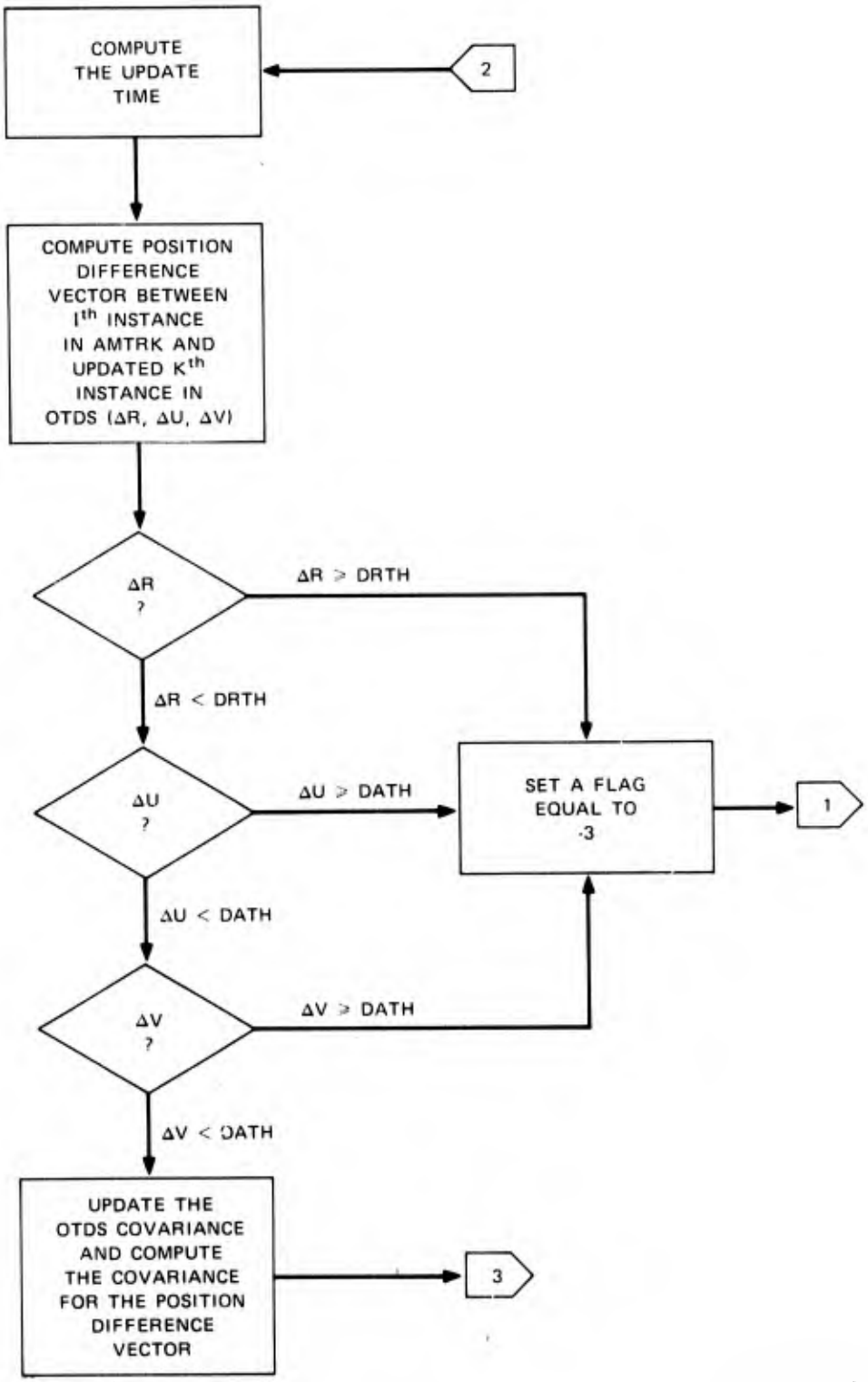


FIGURE C-3 QUECOR FLOW CHART (Continued)

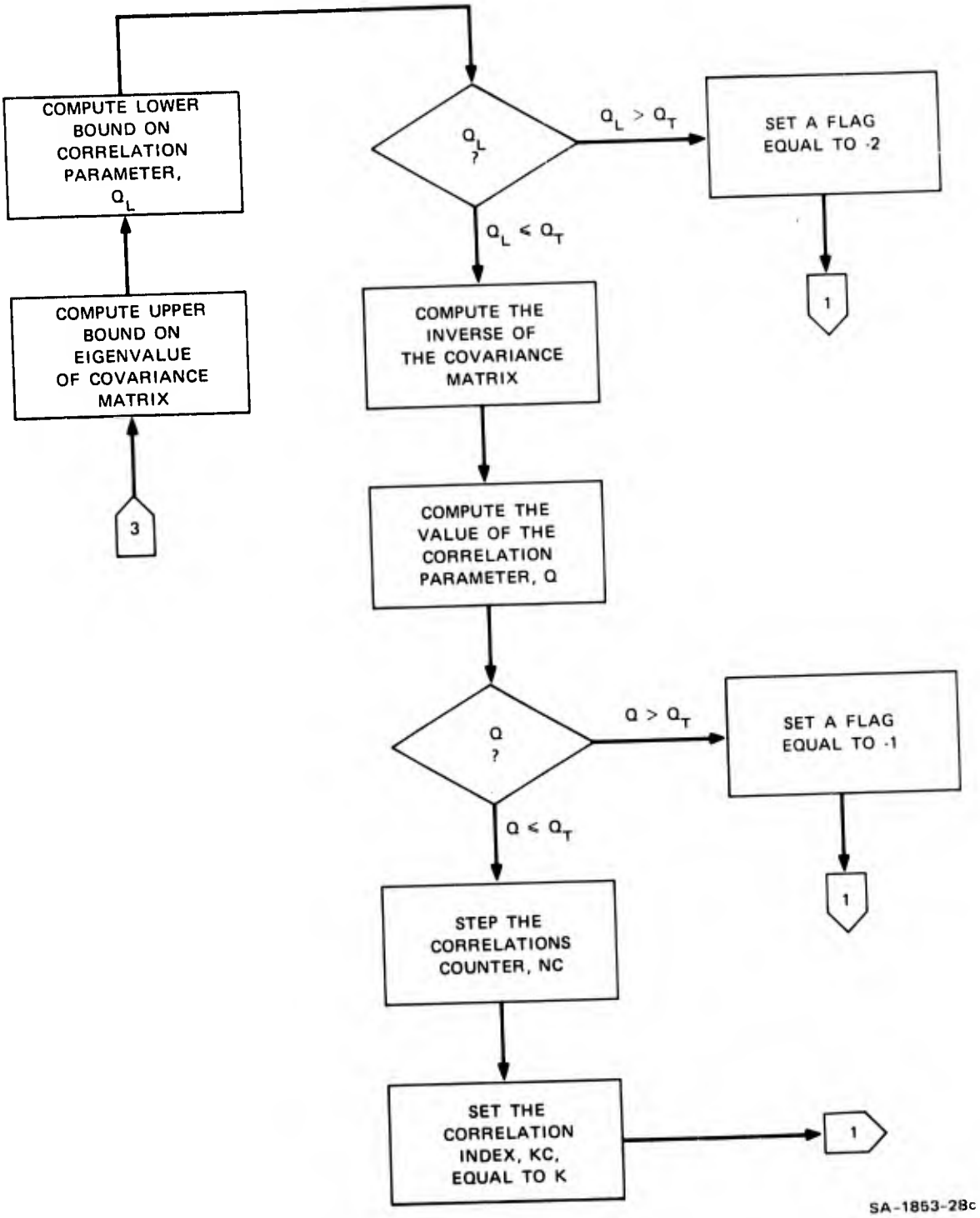


FIGURE C-3 QUECOR FLOW CHART (Concluded)

These data are obtained sequentially for each instance in the OTDS file and are processed before the next instance is obtained. An instance in the OTDS corresponds to a separate track. At this point QUECOR records the OTDS instance.

The processing beyond this point is repeated for each pair of tracks, consisting of the previously selected AMRAD track and currently loaded HAPDAR track.

Since the two times of validity for the two tracks do not generally coincide, the HAPDAR state is updated to the time of validity for the AMRAD track. During this process, the difference vector between the two state vectors is obtained. Also during this process, a coarse screening takes place as each component of the difference vector is computed. This coarse screening consists of a gross check of each of the components of the difference vector. If any component is so large that the marginal probability of the objects being correlated is sufficiently small, we can infer that Q will exceed the threshold and there is no need to actually compute the value of Q. When this happens, a flag for the current HAPDAR track being processed is set to -3, processing for the current track pair is terminated, and the next OTDS instance data is loaded into the buffer and processed. Derivation of appropriate thresholds for the coarse screening was to have been accomplished based on experience with the correlation algorithm. This was not accomplished due to the paucity of good experimental data.

Should the processing pass the coarse screening tests, it is next determined whether a polynomial or Kalman track is being processed. In either case, the HAPDAR covariance is updated to the AMRAD time and the sum of the two covariances is obtained. However, if we have a Kalman filter, the AMRAD covariance must first be transformed to the (X, Y, Z) coordinate system, as must the difference vector previously obtained.

The next step in QUECOR processing is to apply a second level of screening, called prescreening. At this level, a lower bound on Q is computed using Eqs. (9) and (10). Again, should this lower bound be greater than the threshold, there is no need to actually compute Q. When this happens, the flag is set to -2, processing for the current track pair is terminated, and the next OTDS instance data is loaded into the buffer and processed.

Should the processing pass the prescreening test, the value of Q is computed using Eq. (2). At this point, if Q exceeds the threshold, the flag is set equal to -1; otherwise the flag is set equal to 0. Also, a correlation counter is stepped up to count the number of correlations for the handed-over track, and the identification number of the correlated OTDS track is saved. This completes the processing for the current track pair, and the next OTDS instance data is then loaded into the buffer and processed.

The above processing continues until all instances in the OTDS file are exhausted. When this finally occurs, the output data that is contained in the data file, QDATA, is recorded.

QDATA contains the following:

(1) An identification number for the AMRAD track just processed (either 1 or 2 for the first or second AMRAD track).

(2) The correlation time (corresponds to the AMRAD track data time of validity).

(3) The number of HAPDAR tracks that correlate.

(4) The index for the array of HAPDAR track identifiers that corresponds to the track that correlates. (In case more than one correlates, the index will be for the last track to correlate.)

(5) An array of HAPDAR track identifiers. (These correspond to addresses and serve as unique identifiers.)

(6) An array of numbers corresponding to the identifier array that give the total number of track pulses sent out for each HAPDAR track. (These numbers serve to identify when a HAPDAR track is purged and a new track is initiated with the same identifying address.)

(7) An array of Q values computed during the processing. (These entries are either the lower bound for Q, the calculated value of Q, or when a track pair is coarse-screened, an arbitrary number of 1,000,000.)

(8) An array of flags to indicate the correlation result for each track pair.

Currently QUECOR is set up to process up to 25 HAPDAR tracks and 2 AMRAD tracks.

At this point QUECOR processing returns control of the CPU to TRANFT, which immediately calls SRIBLD.

C. QUECOR Execution Time

Timing runs were made on the CDC 6400 at SRI to determine the execution time of QUECOR, with the following results.

When processing a given pair of tracks (one from each of two radars), three possible paths may be executed within QUECOR when the tracks are uncorrelated. The decorrelation may be determined by coarse screening, by prescreening, or by actual computation of the correlation parameter, Q. Coarse screening entails testing the estimated range and angle separation against a threshold. Prescreening entails computing a lower bound on Q and testing against the Q threshold. Finally, if these two

tests are passed, the value of Q is computed and tested against the Q threshold. When the pair of objects are correlated only one path is followed, namely the computation of Q path. Thus, execution times will be a function of the number of times each type of path is followed.

In addition there will be some overhead costs in time due to various initializations and other computations that are performed only once each time QUECOR is called. A part of this overhead cost depends on whether a state and covariance coordinate transformation is required. This requirement is a function of whether any of the tracks are Kalman filtered. If there is at least one Kalman filter then the transformation is required.

To find the execution time let the following definitions hold:

k = number of decorrelations by coarse screening

p = number of decorrelations by prescreening

n = number of decorrelations and correlations by Q calculation

s = 0 or 1 (0 implies no Kalmans and 1 implies at least one Kalman)

t = QUECOR execution time in milliseconds.

The execution time can be obtained from the following equation.

$$t = 0.74k + 0.93p + 1.31n + 0.62s \quad . \tag{13}$$

As consider the case where there are no Kalmans, there are five track pairs to be processed, and the object spacings are such that n = 1, p = 2, and k = 2; then we find that t = 5.4 ms. If there is at least one Kalman, then t = 6.02 ms. In a worst case, where s = 1, n = 5, and both k and p are equal to zero, we obtain t = 7.92 ms.

Only an approximate estimate of the execution time of QUECOR on the IBM 360/65 at HAPDAR was made during the real time system development. This was for a case where n, as defined above, was equal to 3, and k, p, and s were equal to zero. The execution time estimate was 4.5 ms. Using Eq. (13), we obtain 4.68 ms. Thus, it appears that execution times on the CDC and IBM machines are comparable.

A LISTING OF SUBROUTINE QUECOR

```
ISN 0002          SUBROUTINE QUECOR                                                    00001000
                C QUECOR DETERMINES THE CORRELATION OF THE LATEST AMRAD TRACK TO      00002000
                C EACH OF THE HAPDAR TRACKS.                                         00003000
ISN 0003          COMMON/AMTRK/ST1(9),S1,T1,S11,S21,I11,I21,ERR1(10,10),ST2(9),      00004000
                1     S2,T2,S12,S22,I12,I22,ERR2(10,10)                              00005000
ISN 0004          COMMON/QDATA/IDENT,DUMMY,T,NC,KC,ID(25),IQ(25),Q(25),KQ(25)        00005500
ISN 0005          DOUBLE PRECISION ST1,S1,T1,ERR1,ST2,S2,T2,ERR2                     00006000
ISN 0006          INTEGER S11,S21,S12,S22                                            00007000
ISN 0007          DOUBLE PRECISION ERRH(7,7),STH(6),STHK(7),TH,WW                    00008000
ISN 0008          DOUBLE PRECISION T                                                 00008500
ISN 0009          DIMENSION ST(6),ERR(6,6),STX(3),ERX(3,3)                           00009000
ISN 0010          INTEGER DUMMY                                                      00009500
                C                                                                    00010000
ISN 0011        3 FORMAT (/6X,13H TRACK NO. = ,I5,7H ID  = ,Z10,11H ID FLAG = ,I5,9H 00011000
                1 TIME  = ,G15.7)                                                    00012000
ISN 0012        4 FORMAT (//13H AMRAD INPUT //)                                      00013000
ISN 0013        5 FORMAT (/6X,7H ID  = ,I5,9H TIME  = ,G15.7)                        00014000
ISN 0014        6 FORMAT (/6X,10H STATE  = ,3G15.7/6X,15H COVARIANCE  = /)           00015000
ISN 0015        8 FORMAT (//15H HAPDAR INPUTS //)                                    00016000
ISN 0016       10 FORMAT (/6X,15H STATE(RUV)  = /6G15.7)                             00017000
ISN 0017       11 FORMAT (/6X,15H STATE(XYZ)  = /7G15.7/6X,15H COVARIANCE  = /)      00018000
ISN 0018       12 FORMAT (6X,6G15.7)                                                 00019000
ISN 0019       13 FORMAT (//21H CORRELATION RESULTS //)                              00020000
ISN 0020       15 FORMAT(/6X,17H NO. OF CORR.  = ,I5,18H CORR. OBJ. ID  = ,Z10)      00021000
ISN 0021       16 FORMAT (/6X,12H Q ARRAY  = /25G10.1)                               00022000
ISN 0022       17 FORMAT (/6X,17H Q FLAG ARRAY  = /25I10)                            00023000
ISN 0023       18 FORMAT (/6X,13H ID ARRAY  = /25Z10)                                00024000
ISN 0024       19 FORMAT (/6X,18H ID FLAG ARRAY  = /25I10)                           00025000
                C     CHECK FOR INVALID DATA IN AMTRK                                00025100
ISN 0025          IF (T2 .GE. T1) GO TO 21                                           00025200
ISN 0027          IF (ERR1(1,1) .GE. 10000000.0) RETURN                              00025300
ISN 0029          IF (ERR1(2,2) .GT. 1.0) RETURN                                     00025400
ISN 0031          IF (ERR1(3,3) .GT. 1.0) RETURN                                     00025500
ISN 0033          GO TO 22                                                           00025550
ISN 0034       21 CONTINUE                                                           00025600
ISN 0035          IF (ERR2(1,1) .GE. 10000000.0) RETURN                              00025700
ISN 0037          IF (ERR2(2,2) .GT. 1.0) RETURN                                     00025800
ISN 0039          IF (ERR2(3,3) .GT. 1.0) RETURN                                     00025900
ISN 0041       22 CONTINUE                                                           00025950
                C     INITIALIZATIONS                                                00026000
ISN 0042          DATA OUT/0.0/                                                      00026500
ISN 0043          IF (OUT .EQ. 0.0) GO TO 96                                         00026600
ISN 0045          DUMMY = 0                                                          00026700
```

```
ISN 0046            KPRINT=0                                                00027000
ISN 0047            INADD = 0                                               00028000
ISN 0048            QTH = 1.0E60                                            00029000
ISN 0049            DRTH = 1.0E60                                           00030000
ISN 0050            DATH = 1.0E60                                           00031000
ISN 0051            NC = 0                                                  00032000
ISN 0052            KFLAG = 0                                               00033000
ISN 0053            B1 = 0.0                                                00034000
ISN 0054            B2 = 0.0                                                00035000
ISN 0055            B3 = 0.0                                                00036000
ISN 0056            B4 = 0.0                                                00037000
ISN 0057            B5 = 0.0                                                00038000
ISN 0058            B6 = 0.0                                                00039000
                C   RECORD AMTRK                                            00039400
ISN 0059            CALL WRTFIL('AMTRK',IGETA(ST1(1)),1)                    00039500
                C   LOAD AMRAD STATE AND COVARTANCE                        00040000
ISN 0060            IF (T2 .GE. T1) GO TO 25                                00041000
ISN 0062            IDENT = 1                                               00042000
ISN 0063            T = T1                                                  00043000
ISN 0064            ST(1) = ST1(1)                                          00044000
ISN 0065            ST(2) = ST1(2)                                          00045000
ISN 0066            ST(3) = ST1(3)                                          00046000
ISN 0067            ERR(1,1) = ERR1(1,1)                                    00047000
ISN 0068            ERR(2,2) = ERR1(2,2)                                    00048000
ISN 0069            ERR(3,3) = ERR1(3,3)                                    00049000
ISN 0070            ERR(1,2) = ERR1(1,2)                                    00050000
ISN 0071            ERR(2,3) = ERR1(2,3)                                    00051000
ISN 0072            ERR(1,3) = ERR1(1,3)                                    00052000
ISN 0073            GO TO 35                                                00053000
ISN 0074         25 IDENT = 2                                               00054000
ISN 0075            T = T2                                                  00055000
ISN 0076            ST(1) = ST2(1)                                          00056000
ISN 0077            ST(2) = ST2(2)                                          00057000
ISN 0078            ST(3) = ST2(3)                                          00058000
ISN 0079            ERR(1,1) = ERR2(1,1)                                    00059000
ISN 0080            ERR(2,2) = ERR2(2,2)                                    00060000
ISN 0081            ERR(3,3) = ERR2(3,3)                                    00061000
ISN 0082            ERR(1,2) = ERR2(1,2)                                    00062000
ISN 0083            ERR(2,3) = ERR2(2,3)                                    00063000
ISN 0084            ERR(1,3) = ERR2(1,3)                                    00064000
ISN 0085         35 CONTINUE                                                00065000
ISN 0086            IF (KPRINT .EQ. 0) GO TO 45                             00066000
ISN 0088            ERR(2,1) = ERR(1,2)                                     00067000
ISN 0089            ERR(3,1) = ERR(1,3)                                     00068000
ISN 0090            ERR(3,2) = ERR(2,3)                                     00069000
ISN 0091            PRINT 4                                                 00070000
ISN 0092            PRINT 5, IDENT,T                                        00071000
ISN 0093            PRINT 6, (ST(I),I=1,3)                                  00072000
```

```
ISN 0094            DO 65 I = 1,3                                                00073000
ISN 0095          65 PRINT 12, (ERR(I,J),J=1,3)                                  00074000
ISN 0096            PRINT 8                                                      00075000
ISN 0097          45 CONTINUE                                                    00076000
                C     ADD IN BIAS TERMS TO COVARIANCE                            00077000
ISN 0098            ERR(1,1) = B1 + ERR(1,1)                                     00078000
ISN 0099            ERR(2,2) = B2 + ERR(2,2)                                     00079000
ISN 0100            ERR(3,3) = B3 + ERR(3,3)                                     00080000
ISN 0101            ERR(1,2) = B4 + ERR(1,2)                                     00081000
ISN 0102            ERR(2,3) = B5 + ERR(2,3)                                     00082000
ISN 0103            ERR(1,3) = B6 + ERR(1,3)                                     00083000
ISN 0104            I = 0                                                        00084000
ISN 0105          50 CONTINUE                                                    00085000
                C     LOAD, SEQUENTIALLY, HAPDAR STATE AND COVARIANCE            00086000
ISN 0106            CALL GETOTD (ERRH,STH,TH,STHK,WW,IDD,IM,IH,INADD)            00087000
ISN 0107            IF (INADD .EQ. 0) GO TO 150                                  00088000
                C     RECORD HAPDAR STATE AND COVARIANCE                         00088400
ISN 0109            CALL WRTFIL('OTDS',INADD,46)                                 00088500
ISN 0110            I = I + 1                                                    00089000
ISN 0111            IF (STHK(7) .NE. -3000000) GO TO 76                          00089100
ISN 0113            IF (DUMMY .EQ. 0) DUMMY = I                                  00089200
ISN 0115            IF (DUMMY .NE. I) DUMMY = DUMMY + 100000*I                   00089300
ISN 0117          76 CONTINUE                                                    00089400
ISN 0118            IQ(I) = IM + IH                                              00090000
ISN 0119            ID(I) = IDD                                                  00091000
ISN 0120            IF (KPRINT .EQ. 0) GO TO 55                                  00092000
ISN 0122            PRINT 3, I,IDD,IQ(I),TH                                      00093000
ISN 0123            PRINT 10, (STH(J),J=1,6)                                     00094000
ISN 0124            PRINT 11, (STHK(J),J=1,7)                                    00095000
ISN 0125            DO 75 J = 1,6                                                00096000
ISN 0126          75 PRINT 12, (ERRH(J,K),K=1,6)                                 00097000
ISN 0127          55 CONTINUE                                                    00098000
ISN 0128            KQ(I) = 0                                                    00099000
ISN 0129            DT = T - TH                                                  00100000
ISN 0130            DTSQ = DT*DT                                                 00101000
ISN 0131            DT2 = 2*DT                                                   00102000
                C     UPDATE HAPDAR STATE AND COVARIANCE TO TIME OF VALIDITY FOR AMRAD 00103000
                C     STATE  IN CASE OF POLY. FILTER                             00104000
ISN 0132            D1 = ST(1) - STH(1) - DT*STH(4)                              00105000
ISN 0133            IF (ABS(D1) .GT. DRTH) GO TO 85                              00106000
ISN 0135            D2 = ST(2) - STH(2) - DT*STH(5)                              00107000
ISN 0136            IF (ABS(D2) .GT. DATH) GO TO 85                              00108000
ISN 0138            D3 = ST(3) - STH(3) - DT*STH(6)                              00109000
ISN 0139            IF (ABS(D3) .GT. DATH) GO TO 85                              00110000
                C     DETERMINE IF KALMAN OR POLYNOMIAL FILTER                   00111000
ISN 0141            IF (STHK(7) .GT. -1000000) GO TO 250                         00112000
                C     CONTINUE ON TO UPDATE OF HAPDAR COVARIANCE  IN CASE OF POLY. FIL. 00113000
ISN 0143            C1 = ERR(1,1) + ERRH(1,1) + DT2*ERRH(1,4) + DTSQ*ERRH(4,4)   00114000
```

```
ISN 0144            C2 = ERR(2,2) + ERRH(2,2) + DT2*ERRH(2,5) + DTSQ*ERRH(5,5)          00115000
ISN 0145            C3 = ERR(3,3) + ERRH(3,3) + DT2*ERRH(3,6) + DTSQ*ERRH(6,6)          00116000
ISN 0146            C4 = ERR(1,2)                                                       00117000
ISN 0147            C5 = ERR(2,3)                                                       00118000
ISN 0148            C6 = ERR(1,3)                                                       00119000
ISN 0149            GO TO 350                                                           00120000
ISN 0150          250 KFLAG = KFLAG + 1                                                 00121000
ISN 0151            IF (KFLAG .NE. 1) GO TO 450                                         00122000
                C   TRANSFORM AMRAD RUV SYSTEM TO HAPDAR XYZ SYSTEM  IN CASE AT LEAST  00123000
                C   ONE KALMAN FILTER                                                  00124000
ISN 0153            US = ST(2)*ST(2)                                                    00125000
ISN 0154            VS = ST(3)*ST(3)                                                    00126000
ISN 0155            WS = ABS(1.0 - US - VS)                                             00127000
ISN 0156            RS = ST(1)*ST(1)                                                    00128000
ISN 0157            UV = ST(2)*ST(3)                                                    00129000
ISN 0158            W = SQRT(WS)                                                        00130000
ISN 0159            UW = W*ST(2)                                                        00131000
ISN 0160            VW = W*ST(3)                                                        00132000
ISN 0161            RW = ST(1)/W                                                        00133000
ISN 0162            RWS = RW*RW                                                         00134000
ISN 0163            X = ST(1)*ST(2)                                                     00135000
ISN 0164            Y = ST(1)*ST(3)                                                     00136000
ISN 0165            WMU = WS - US                                                       00137000
ISN 0166            WMV = WS - VS                                                       00138000
ISN 0167            ERX(1,1)= US*ERR(1,1) + RS*ERR(2,2) + 2*X*ERR(1,2)                  00139000
ISN 0168            ERX(2,2) = VS*ERR(1,1) + RS*ERR(3,3) + 2*Y*ERR(1,3)                 00140000
ISN 0169            ERX(3,3) = WS*ERR(1,1) - 2*X*ERR(1,2) - 2*Y*ERR(1,3)                00141000
ISN 0170            ERX(3,3) = ERX(3,3)+RWS*(US*ERR(2,2)+VS*ERR(3,3)+2*UV*ERR(2,3))     00142000
ISN 0171            ERX(1,2) = UV*ERR(1,1) + Y*ERR(1,2) + X*ERR(1,3) + RS*ERR(2,3)      00143000
ISN 0172            ERX(1,3) = WMU*ERR(1,2) - X*ERR(2,2) - UV*ERR(1,3) - Y*ERR(2,3)     00144000
ISN 0173            ERX(1,3) = ERX(1,3)*RW + UW*ERR(1,1)                                00145000
ISN 0174            ERX(2,3) = WMV*ERR(1,3) - Y*ERR(3,3) - UV*ERR(1,2) - X*ERR(2,3)     00146000
ISN 0175            ERX(2,3) = ERX(2,3)*RW + VW*ERR(1,1)                                00147000
ISN 0176            STX(1) = X                                                          00148000
ISN 0177            STX(2) = Y                                                          00149000
ISN 0178            STX(3) = ST(1)*W                                                    00150000
ISN 0179          450 CONTINUE                                                          00151000
                C   UPDATE HAPDAR STATE AND COVARIANCE TO TIME OF VALIDITY FOR AMRAD   00152000
                C   STATE  IN CASE OF KALMAN FILTER                                    00153000
ISN 0180            D1 = STX(1) - STHK(1) - DT*STHK(4)                                  00154000
ISN 0181            D2 = STX(2) - STHK(2) - DT*STHK(5)                                  00155000
ISN 0182            D3 = STX(3) - STHK(3) - DT*STHK(6)                                  00156000
ISN 0183            C1 = ERX(1,1) + ERRH(1,1) + DT2*ERRH(1,4) + DTSQ*ERRH(4,4)          00157000
ISN 0184            C2 = ERX(2,2) + ERRH(2,2) + DT2*ERRH(2,5) + DTSQ*ERRH(5,5)          00158000
ISN 0185            C3 = ERX(3,3) + ERRH(3,3) + DT2*ERRH(3,6) + DTSQ*ERRH(6,6)          00159000
ISN 0186            C4 = ERX(1,2)+ERRH(1,2)+DT*(ERRH(1,5)+ERRH(4,2))+DTSQ*ERRH(4,5)     00160000
ISN 0187            C5 = ERX(2,3)+ERRH(2,3)+DT*(ERRH(2,6)+ERRH(5,3))+DTSQ*ERRH(5,6)     00161000
ISN 0188            C6 = ERX(1,3)+ERRH(1,3)+DT*(ERRH(1,6)+ERRH(4,3))+DTSQ*ERRH(4,6)     00162000
```

```
ISN 0189        350 CONTINUE                                                    00163000
                C       COMPUTE LOWER BOUND ON CORRELATION PARAMETER, Q         00164000
ISN 0190                DSQ = D1*D1 + D2*D2 + D3*D3                             00165000
ISN 0191                AC4 = ABS(C4)                                           00166000
ISN 0192                AC5 = ABS(C5)                                           00167000
ISN 0193                AC6 = ABS(C6)                                           00168000
ISN 0194                SIGM = C1 + AC4 + AC6                                   00169000
ISN 0195                TEMP = C2 + AC4 + AC5                                   00170000
ISN 0196                IF (TEMP .GT. SIGM) SIGM = TEMP                         00171000
ISN 0198                TEMP = C3 + AC6 + AC5                                   00172000
ISN 0199                IF (TEMP .GT. SIGM) SIGM = TEMP                         00173000
ISN 0201                IF (SIGM .GT. 0.0) GO TO 1000                           00173100
ISN 0203                KQ(I) = -4                                              00173200
ISN 0204                GO TO 50                                                00173300
ISN 0205        1000 CONTINUE                                                   00173400
ISN 0206                Q(I) = DSQ/SIGM                                         00174000
ISN 0207                IF (Q(I) .LE. QTH) GO TO 100                            00175000
ISN 0209                KQ(I) = -2                                              00175000
ISN 0210                GO TO 50                                                00177000
ISN 0211         100 CONTINUE                                                   00178000
                C       COMPUTE CORRELATION PARAMETER, Q  IN CASE LOWER BOUND LESS THAN  00179000
                C       THRESHOLD                                               00180000
ISN 0212                CI1 = C2*C3 - C5*C5                                     00181000
ISN 0213                CI2 = C1*C3 - C5*C6                                     00182000
ISN 0214                CI3 = C1*C2 - C4*C4                                     00183000
ISN 0215                CI4 = C6*C5 - C4*C3                                     00184000
ISN 0216                CI5 = C4*C6 - C1*C5                                     00185000
ISN 0217                CI6 = C4*C5 - C6*C2                                     00186000
ISN 0218                TEMP = CI3*C3 + CI5*C5 + CI6*C6                         00187000
ISN 0219                IF (TEMP .GT. 0.0) GO TO 1100                           00187100
ISN 0221                KQ(I) = -5                                              00187200
ISN 0222                GO TO 50                                                00187300
ISN 0223        1100 CONTINUE                                                   00187400
ISN 0224                Q(I) = CI1*D1*D1 + CI2*D2*D2 + CI3*D3*D3                00188000
ISN 0225                Q(I) =(Q(I) + 2*(CI4*D1*D2 + CI5*D2*D3 + CI6*D1*D3))/TEMP  00189000
ISN 0226                IF (Q(I) .GT. QTH) GO TO 200                            00190000
ISN 0228                NC = NC + 1                                             00191000
ISN 0229                KC = ID(I)                                              00192000
ISN 0230                GO TO 50                                                00193000
ISN 0231         200 KQ(I) = -1                                                 00194000
ISN 0232                GO TO 50                                                00195000
ISN 0233          85 KQ(I) = -3                                                 00196000
ISN 0234                Q(I) = 1000000                                          00197000
ISN 0235                GO TO 50                                                00198000
ISN 0236         150 CONTINUE                                                   00199000
ISN 0237                IMAX = I                                                00200000
ISN 0238                ID(I+1) = 0                                             00200500
ISN 0239                IF (KPRINT .EQ. 0) GO TO 95                             00201000
```

```
ISN 0241            PRINT 13                                              00202000
ISN 0242            PRINT 5, IDENT,T                                      00203000
ISN 0243            PRINT 15, NC,KC                                       00204000
ISN 0244            PRINT 18, (ID(J),J=1,IMAX)                            00205000
ISN 0245            PRINT 19, (IQ(J),J=1,IMAX)                            00206000
ISN 0246            PRINT 16, (Q(J),J=1,IMAX)                             00207000
ISN 0247            PRINT 17, (KQ(J),J=1,IMAX)                            00208000
ISN 0248         95 CONTINUE                                              00209000
                 C  RECORD CORRELATION RESULTS                            00209400
ISN 0249            CALL WRTFIL('QDATA',IGETA(IDENT),1)                   00209500
ISN 0250         96 CONTINUE                                              00209600
ISN 0251            OUT = 100.0                                           00209700
ISN 0252            RETURN                                                00210000
ISN 0253            END                                                   00211000
```

Appendix D

SPECIAL SEARCH PROGRAM SSERCH

Appendix D

SPECIAL SEARCH PROGRAM SSERCH

SSERCH is a real time, field test algorithm that generates a series of beam pointing directions and range gates required for HAPDAR to search a volume on special request. This might be the shadowed volume behind a nuclear fireball that temporarily blocks out part of another radar's search volume (in our case AMRAD). The purpose of this appendix is to briefly describe the logical structure and operation of SSERCH, and to present a listing of SSERCH.

SSERCH was installed in the tactical function library in the IBM 360/65 at the HAPDAR facility. SSERCH was never integrated into the PHSD system due to the priority on the handover and correlation software integration and testing and subsequent delays in achieving that task. Thus, SSERCH has never been validated in real time operation.

SSERCH was meant to be executed periodically, once per second, whenever QUECOR is executed. The inputs required by SSERCH are contained in two data files, AMTRK and SSDATA. AMTRK contains the two AMRAD track data sets, and SSDATA contains the results in terms of hits or misses for the previously requested search beams. SSDATA also contains the output data of SSERCH, which consists of the requested search beam pointing information.

SSERCH assumes that the desired search volume consists of the region in the shadow of a hypothetical fireball of a fixed radius located at a range equal to a fixed distance in front of the handed over state vector point obtained from AMTRK. The hypothetical fireball is assumed to lie on the line-of-sight from AMRAD to the handed over point, and the conical

shadow region is defined relative to AMRAD. The search volume is assumed to extend from the front edge of the fireball to a maximum distance behind the fireball.

A flow chart for SSERCH is contained in Figure D-1. When SSERCH is called, it first loads data from AMTRK into a buffer. The data obtained consist of the position components (R, u, v) of the first AMRAD track. SSERCH then computes the direction cosines for the handed over point with respect to AMRAD, using the following vector equation;

$$\underset{\sim}{e}_A^T = \left(\underset{\sim}{Re}_H^T - \underset{\sim}{x}_{AH}^T\right) \Big/ \left|\underset{\sim}{Re}_H^T - \underset{\sim}{x}_{AH}^T\right| \tag{14}$$

where

$$\underset{\sim}{e}_H^T = [u,\ v,\ w] \tag{15}$$

$$w = \sqrt{1 - u^2 - v^2} \tag{16}$$

$\underset{\sim}{x}_{AH}^T$ = transpose of the vector locating AMRAD relative to HAPDAR in HAPDAR face centered cartesian coordinates

Next, the range and the direction of the fireball relative to HAPDAR are determined using the relationships

$$R_1 = \left|\underset{\sim}{Re}_H^T - \Delta\underset{\sim}{Re}_A^T\right| \tag{17}$$

and

$$\underset{\sim}{e}_{H1} = \left(\underset{\sim}{Re}_H^T - \Delta\underset{\sim}{Re}_A^T\right) \Big/ R_1 \tag{18}$$

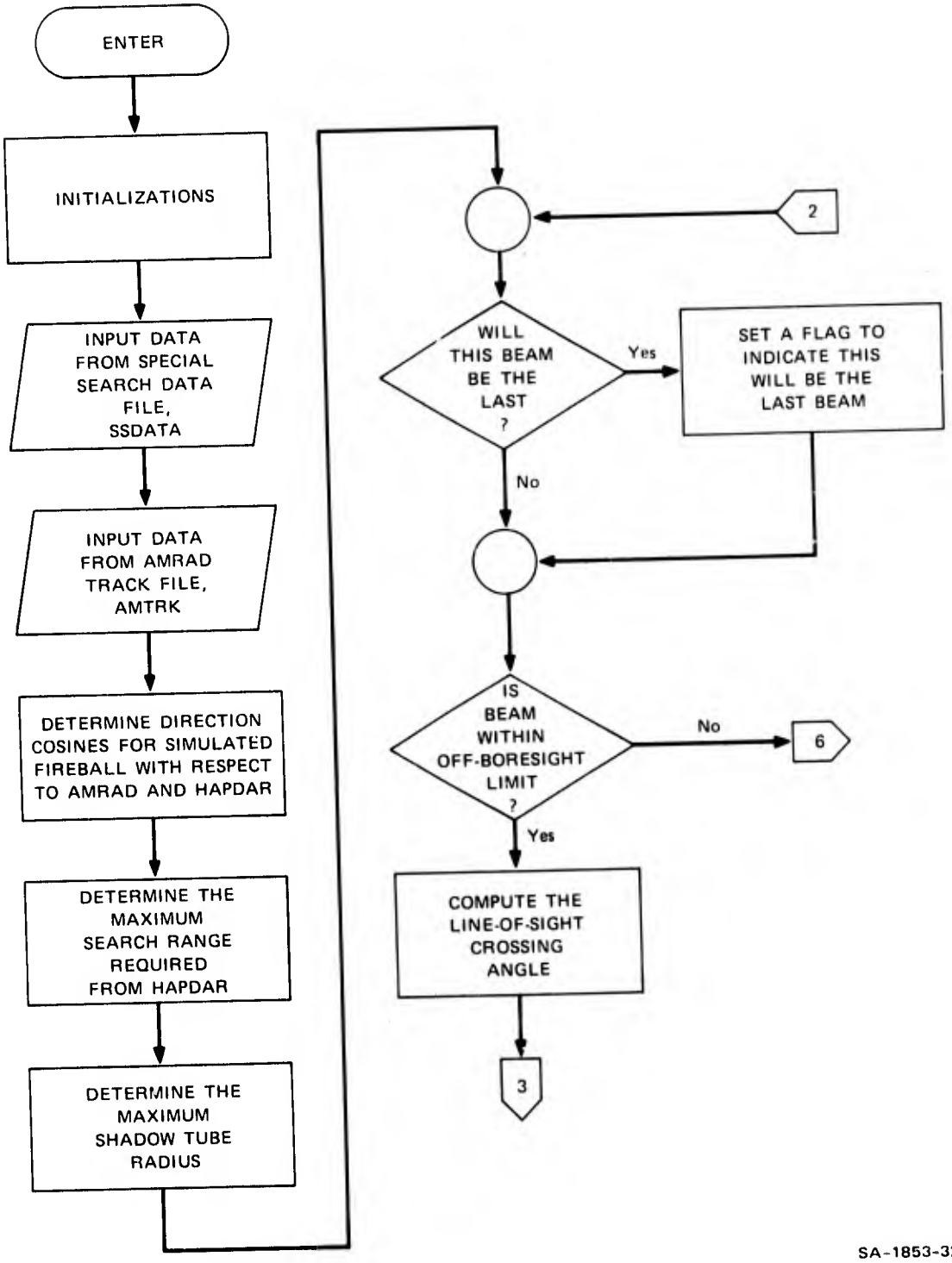


FIGURE D-1 SSERCH FLOW CHART

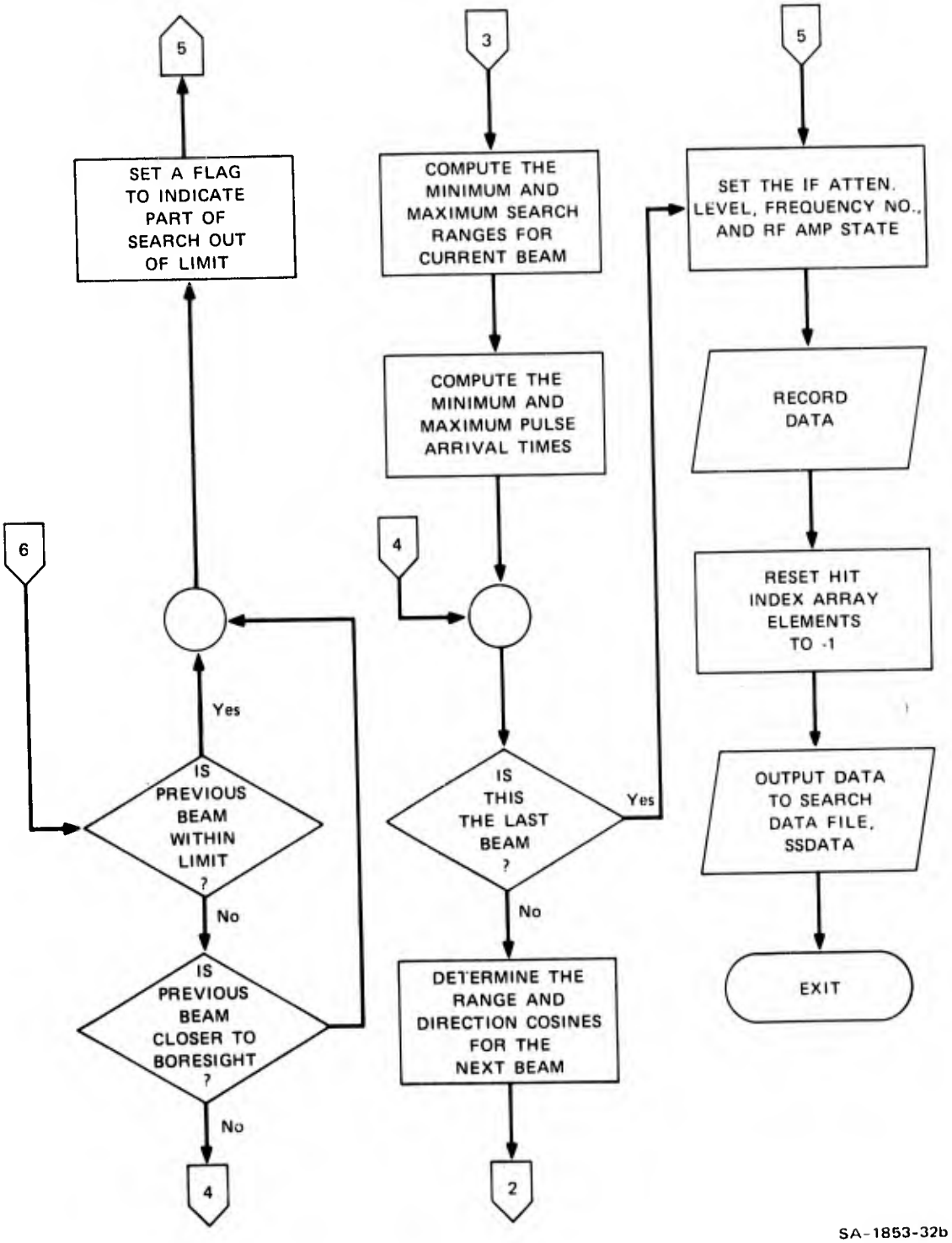


FIGURE D-1 SSERCH FLOW CHART (Concluded)

where

ΔR = the assumed distance of the fireball in front of the handed over point relative to AMRAD.

The subscripts "1" indicate that these are the pointing parameters for the first search beam.

The maximum search range required from HAPDAR is then computed, using the equation

$$R_m = \left| R\underset{\sim}{e}_H^T + \Delta R\underset{\sim}{e}_A^T \right| \quad . \tag{19}$$

Here we assumed for the field tests that the maximum search range relative to AMRAD is ΔR behind the handed over point. Next, the maximum shadow tube radius created by the fireball at the maximum search range is determined:

$$r_{FM} = (R_A + 2\Delta R)\ r_F/R_A \tag{20}$$

where

$$R_A = \left| R_1 \underset{\sim}{e}_{H1}^T - \underset{\sim}{x}_{AH}^T \right| \tag{21}$$

r_F = assumed fireball radius.

Using R_1 and $\underset{\sim}{e}_{H1}$ as starting points, SSERCH successively computes beam pointing parameters for a succession of up to nine beams, which cover the desired search volume. These beam positions are determined so that they sweep along the line-of-sight from AMRAD to the handed over point between the two range limits. Figure D-2 shows schematically the various beams and range gates in the plane containing AMRAD, HAPDAR, and the AMRAD line-of-sight. Figure D-3 shows schematically how the range gate limits are determined.

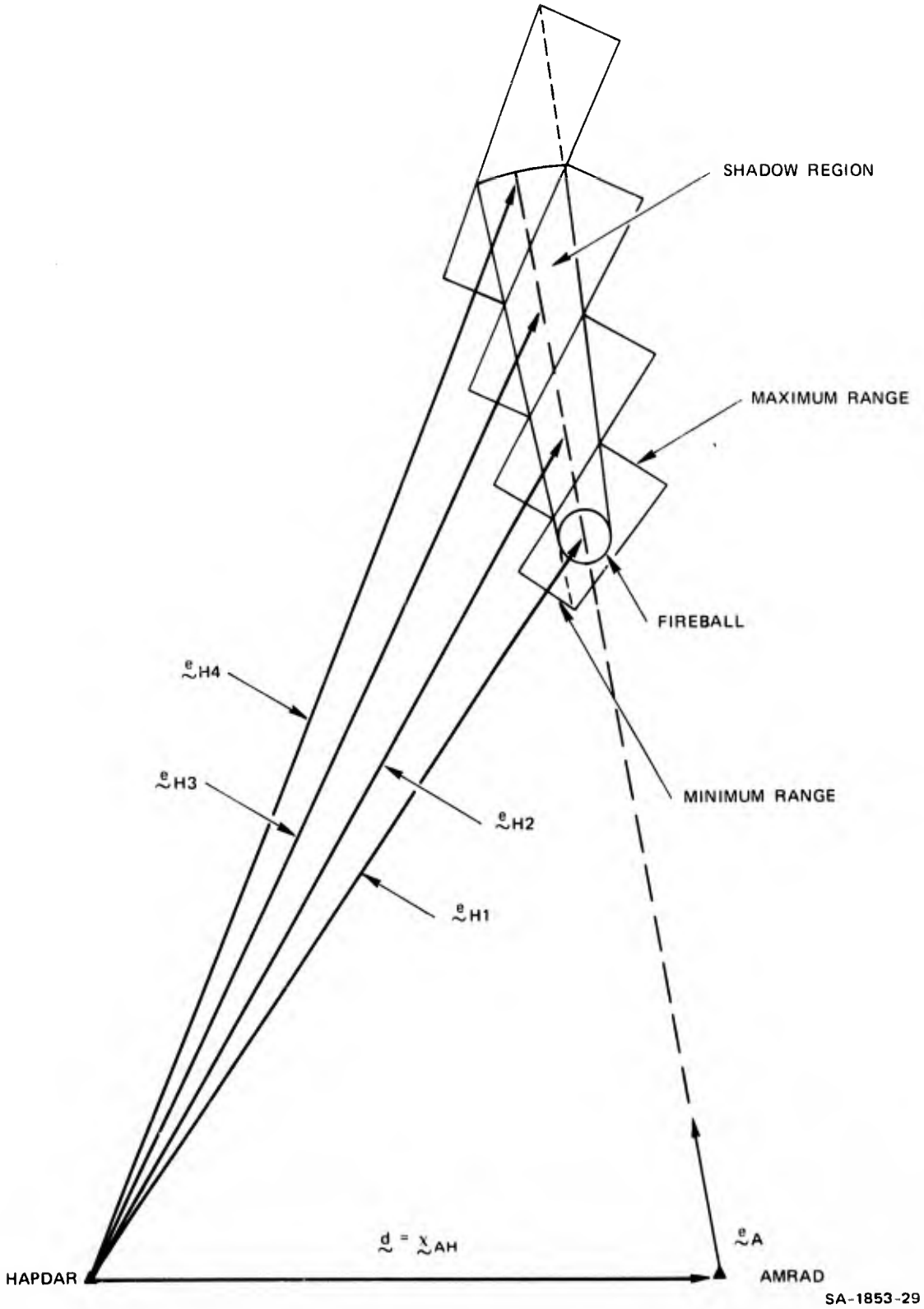


FIGURE D-2 SEARCH BEAM POSITIONS

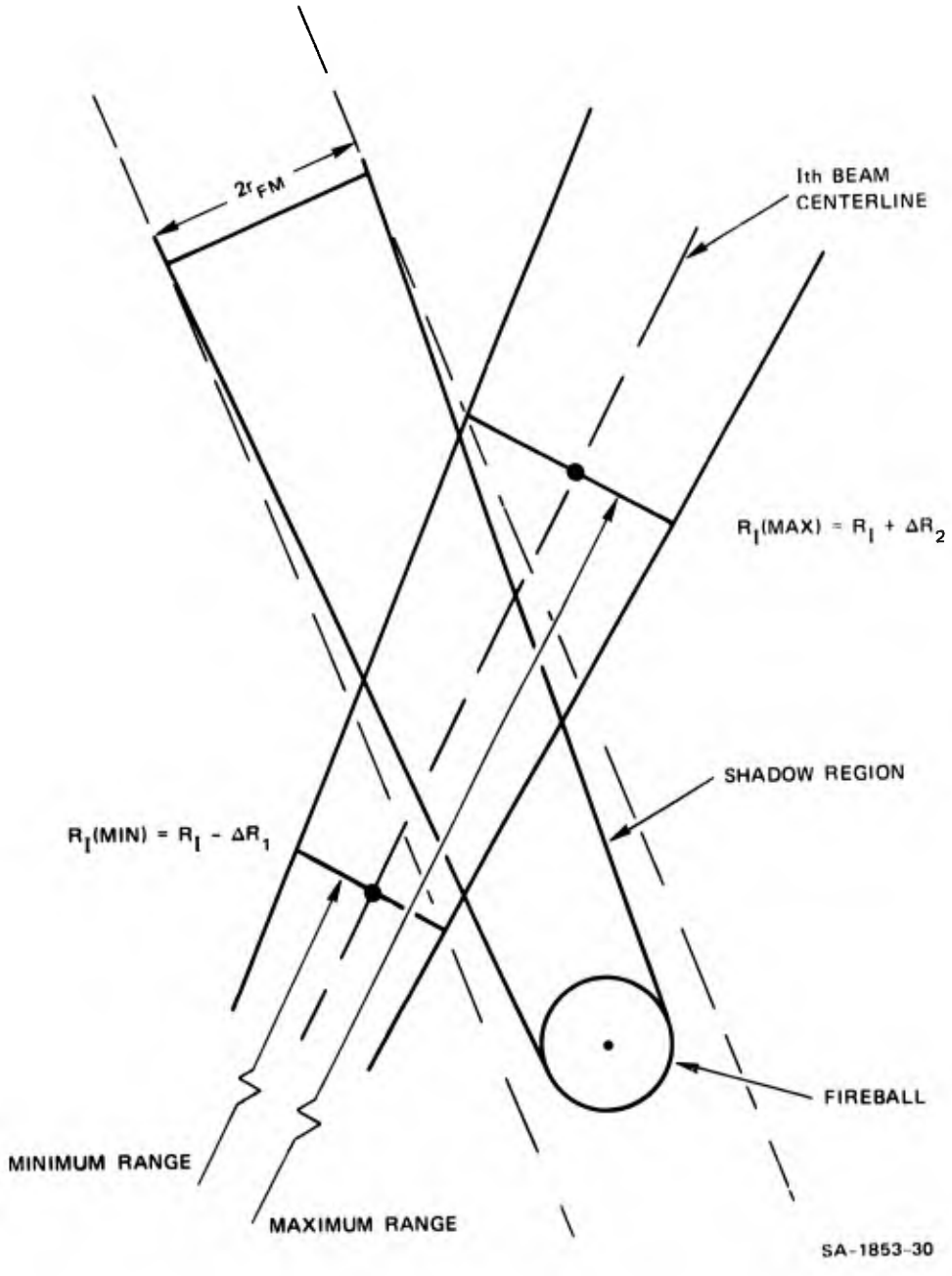


FIGURE D-3 RANGE GATE GEOMETRY

As each beam is generated, SSERCH determines if R_m is exceeded, and if so, a flag is set to indicate that the last beam is being generated. Next, SSERCH determines if the beam is within the maximum off-boresight limits for HAPDAR. If not, then SSERCH determines if any subsequent beams might fall within the off-boresight limits. If all subsequent beams are predicted to fall out of the off-boresight limits, no further beams are generated, and the processing proceeds to the final operations. When this happens, a flag is also set to indicate that at least some part of the requested search volume falls out of limits. If subsequent beams are expected within off-boresight limits, SSERCH next checks the previously set flag to determine if the current beam was to be the last. If so, SSERCH proceeds to the final operations. Otherwise, the next beam pointing parameters are determined.

If the current beam being processed falls within the HAPDAR off-boresight limits, then SSERCH computes the range gate parameters for the beam. The range gates are determined in terms of the pulse arrival times after transmission by the following equations:

$$T_{1I} = 2R_{Imin}/v \tag{22}$$

$$T_{2I} = 2R_{Imax}/v$$

where

$$v = \text{velocity of light}$$

$$R_{Imin} = (R_I \sin\theta \cos\varphi - r_{FM} \tan\theta)/\sin(\theta-\varphi) \tag{24}$$

$$R_{Imax} = (R_I \sin\theta \cos\varphi + r_{FM} \tan\theta)/\sin(\theta-\varphi) \tag{25}$$

$$\varphi = \varphi_o/(2w_I) \tag{26}$$

φ_o = boresight beamwidth of HAPDAR

w_I = off-boresight direction cosine for the I^{th} beam

θ = angle between the AMRAD line-of-sight and the centerline of the I^{th} beam.

The angle θ is given by

$$\cos \theta = \underset{\sim}{e}_A^T \underset{\sim}{e}_{HI} \quad . \tag{27}$$

After the range gates are computed for the I^{th} beam, SSERCH determines if the last required beam has been computed. If so, it proceeds to the final operations; if not, it computes the next beam pointing parameters.

Successive beam pointing directions are determined in u-v space, since in this space the beamwidth is represented by a constant diameter circle. Figure D-4 shows how the next beam is positioned. Based on this positioning scheme, SSERCH next computes the distance, D, along the line-of-sight (from AMRAD) from the I^{th} to the $(I+1)^{th}$ beam. D is given by

$$D = b/a + (b^2/a^2 + c/a)^{1/2} \tag{28}$$

where

$$a = (e_A(1) - e_{HI}(1) \cos \theta)^2 + (e_A(2) - e_{HI}(2) \cos \theta)^2 - 0.1875 \sin^2 \varphi_o \tag{29}$$

$$b = 0.1875 \sin^2 \varphi_o \cos \theta R_I \tag{30}$$

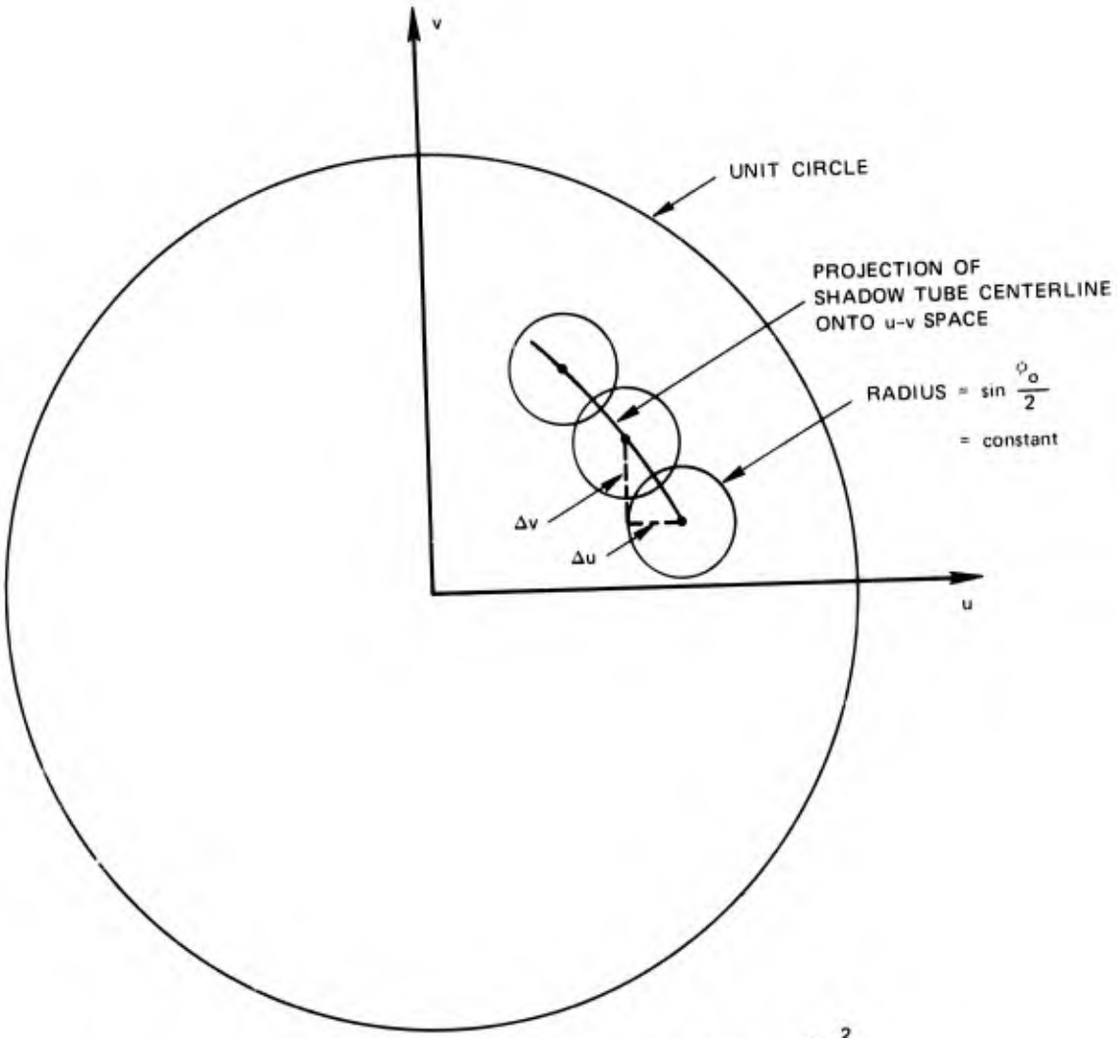


FIGURE D-4 SEARCH BEAM PACKING IN u-v SPACE

$$c = 0.1875 \sin^2 \varphi_o R_I^2 \tag{31}$$

The beam pointing parameters are then computed from

$$R_{I+1} = \left[R_I^2 + 2DR_I \cos \theta + D^2\right]^{1/2} \tag{32}$$

$$\underset{\sim}{e}_{H(I+1)}^T = \left(R_I \underset{\sim}{e}_{HI}^T + D \underset{\sim}{e}_A^T\right) \Big/ R_{I+1} \tag{33}$$

The processing cycles back at this point to the determination of whether this beam will be the last or not. The cycle is repeated until all required beams that are feasible are generated, or until nine beams have been generated, whichever occurs first.

SSERCH then proceeds to the final operation, which consists of selecting the desired IF attenuation level (0 to 63 dB), the frequency number (0, 1, ..., 7), and the RF amplifier state (0 = in, 1 = no preamp, 2 = 30 dB pad, 3 = open), and storing these data in SSDATA. SSERCH then records SSDATA, which contains the results of the previous search request, and the pointing information for the new search request. These data consist of:

(1) The minimum search range

(2) The direction cosine of the search beam relative to the vertical axis in the array face, u

(3) The direction cosine of the search beam relative to the horizontal axis in the array face, v

(4) The minimum pulse arrival time

(5) The maximum pulse arrival time

(6) An IF attenuation level

(7) A frequency index

(8) An RF amplifier state index

(9) Elements of the hit/miss index array reset to -1.

The hit/miss array is a 9 by 10 array whose rows correspond to each requested beam position, and whose columns correspond to the number of search pulses transmitted in a given beam position since the last special search request. An entry of -1 in an element of this array means that no radar beam was formed or transmitted, 0 means no hit or an invalid hit was obtained, 1 through 5 means 1 through 5 hits, respectively, were obtained, and 6 means more than 5 hits were obtained. Entries into this array are made by the search radar return processing function.

This completes SSERCH data processing, and control is returned to the calling routine.

A LISTING OF SUBROUTINE SSERCH

```
                      SUBROUTINE SSERCH
000002                COMMON/AMTRK/ST1(9),S1,T1,S11,S21,I11,I21,ERR1(10,10),ST2(9),
                     1     S2,T2,S12,S22,I12,I22,ERR2(10,10)
000002                DOUBLE PRECISION ST1,S1,T1,ERR1,ST2,S2,T2,ERR2
000002                INTEGER S11,S21,S12,S22
000002                COMMON/SSDATA/DATA(6,9),IDATA(2,9),IHIT(10,9)
000002                DOUBLE PRECISION T,DATA
000002                DIMENSION STATE(6)
                    C**** C = VEL. OF LIGHT (MPS), FBR = FIREBALL RADIUS (M), (XAH,YAH,ZAH)=
                    C**** AMRAD LOCATION IN HAPDAR FACE CENTERED CARTESIAN COORDINATES (M,
                    C**** RAD,RAD), DR = RANGE INTERVAL ABOUT HANDOVER POINT FOR SEARCH (M),
                    C****  AND BW = HAPDAR BEAMWIDTH (RAD)
000002                C = 299792880.0
000004                FBR = 1000.0
000005                XAH = 6393.9
000007                YAH = 10937.27
000010                ZAH = 11050.71
000012                DR = 20000.0
000013                WTH = 0.500
000015                BW = 0.029
000016                SBWSQ = 0.75*BW*BW
000020                IFLAG = 0
000021                KFLAG = 0
000022                N = 0
000023                TT1 = T1
000025                TT2 = T2
000027                IF (TT2 .GT. TT1) GO TO 25
000032                IDENT = 1
000033                T = T1
000035                STATE(1) = ST1(1)
000037                STATE(2) = ST1(2)
000041                STATE(3) = ST1(3)
000043                GO TO 35
000044             25 CONTINUE
000044                IDENT = 2
000045                T = T2
000050                STATE(1) = ST2(1)
000052                STATE(2) = ST2(2)
000054                STATE(3) = ST2(3)
000056             35 CONTINUE
000056                W = ABS(1.0 - STATE(2)*STATE(2) - STATE(3)*STATE(3))
000063                W = SQRT(W)
000065                U = STATE(1)*STATE(2)
000067                V = STATE(1)*STATE(3)
000071                W = STATE(1)*W
000072                EXA = U - XAH
000074                EYA = V - YAH
000076                EZA = W - ZAH
000100                RA = SQRT(EXA*EXA + EYA*EYA + EZA*EZA)
000107                EXA = EXA/RA
000110                EYA = EYA/RA
000111                EZA = EZA/RA
000112                U = U - DR*EXA
000115                V = V - DR*EYA
000120                W = W - DR*EZA
```

```
RUN  VERSION FEB 72 F    17:00 04/16/73

000122          R = SQRT(U*U + V*V + W*W)
000130          U = U/R
000131          V = V/R
000132          W = W/R
000133          WOLD = W
000134          DR2 = 2*DR
000136          RU = R*U + DR2*EXA
000141          RV = R*V + DR2*EYA
000144          RW = R*W + DR2*EZA
000147          RMAX = SQRT(RU*RU + RV*RV + RW*RW)
000156          TR = FBR*(RA + DR)/(RA- DR)
000163          RMX = RMAX + TR
000165          RMN = R - 1.5*TR
000167       10 N = N + 1
000171          IF (R .GT. RMAX) KFLAG = 1
000174          IF (W .LT. WTH) GO TO 20
000177          COSTH = U*EXA + V*EYA + W*EZA
000205          THETA = ACOS(COSTH)
000207          BWMAX = BW/(2*W)
000212          SINTH = SIN(THETA)
000214          COSB = COS(BWMAX)
000216          SINTHP = SIN(THETA + BWMAX)
000222          SINTHM = SIN(THETA - BWMAX)
000226          RR1 = R*SINTH*COSB
000230          RR2 = TR*SINTH/COSTH
000232          R1 = (RR1 - RR2)/SINTHP
000235          IF (R1 .LT. RMX) GO TO 15
000237          N = N - 1
000241          GO TO 50
000241       15 CONTINUE
000241          IF (R1 .LT. RMN) R1 = RMN
000245          DATA(1,N) = R1
000251          R2 = (RR1 + RR2)/SINTHM
000254          IF (R2 .GT. RMX) R2 = RMX
000257          DATA(4,N) = 2*DATA(1,N)/C
000301          DATA(5,N) = 2*R2/C
000306          DATA(2,N) = U
000310          DATA(3,N) = V
000313          DATA(6,N) = 0.0
000315          IDATA(1,N) = 0
000317          IDATA(2,N) = 0
000321          IF (N .EQ. 9) GO TO 40
000323          GO TO 30
000323       20 IF (WOLD  .GE. WTH .OR. WOLD  .GT. W) GO TO 40
000335       30 IF (KFLAG .GT. 0) GO TO 50
000340          A21 = EXA - U*COSTH
000343          A22 = EYA - V*COSTH
000345          A2 = A21*A21 + A22*A22 - SBWSQ
000351          A1 = R*COSTH*SBWSQ
000353          A0 = R*R*SBWSQ
000354          A1 = A1/A2
000355          A0 = A0/A2
000356          DT = ABS(A1*A1 + A0)
000361          DT = A1 + SQRT(DT)
000364          ROLD = R
000366          R = ABS(R*R + 2*DT*R*COSTH + DT*DT)
```

```
RUN  VERSION FEB 72 F   17:00 04/16/73

000374          R = SQRT(R)
000377          U = (ROLD*U + DT*EXA)/R
000403          V = (ROLD*V + DT*EYA)/R
000406          WOLD = W
000410          W = ABS(1.0 - U*U - V*V)
000414          W = SQRT(W)
000417          GO TO 10
000417       40 IFLAG = 1
000420       50 CONTINUE
000420          IMAX = N
000422          DO 60 I = 1,9
000423          DO 60 J = 1,10
000424       60 IHIT(J,I) = -1
            C**** DATA RECORDING STATEMENT REQUIRED HERE TO COMPLETE REAL TIME
            C**** IMPLEMENTATION.
000435          RETURN
000436          END
```

Appendix E

PHSD/SRI HANDOVER SOFTWARE INTERFACE

Appendix E

PHSD/SRI HANDOVER SOFTWARE INTERFACE

The various interfaces among the SRI handover functions, the various data files, and other PHSD tracking functions are shown in Figure E-1. The SRI handover functions consist of VPAS, TRANFT, QUECOR, and SRIBLD. Other interfacing PHSD functions include KRM, ARRP, ARR, DTTSRI, and WLC. The purpose of each of these functions, the data files they access, and the rolling sequence will be briefly described to indicate how the SRI handover software integrates with the PHSD software. (For a more detailed description of the PHSD software see Refs. 4 and 11).

As raw AMRAD data are received over the AMRAD link to HAPDAR, they are loaded into the RSDS data file by VPAS. VPAS essentially provides the function of loading the data from the PAS Return Buffer into the RSDS data file after converting the range time to PHSD mission time and applying a correction of -23 dB to the SNR measured by AMRAD. Right after this is accomplished TRANFT is executed.

The function of TRANFT is to obtain the data from RSDS, convert them to HAPDAR coordinates, and fit them with an appropriate filtering algorithm. TRANFT loads the filtered data into the AMTRK data file. AMTRK contains the filtered state vector, covariance matrix, time of validity, SNR, and status and identification information for the two AMRAD track channels. TRANFT also determines when it is time to request a handover and correlation test. It accomplishes this by calling QUECOR and SRIBLD.

The present version of TRANFT is set up to request a handover once every second on alternately the first and second AMRAD track. New data

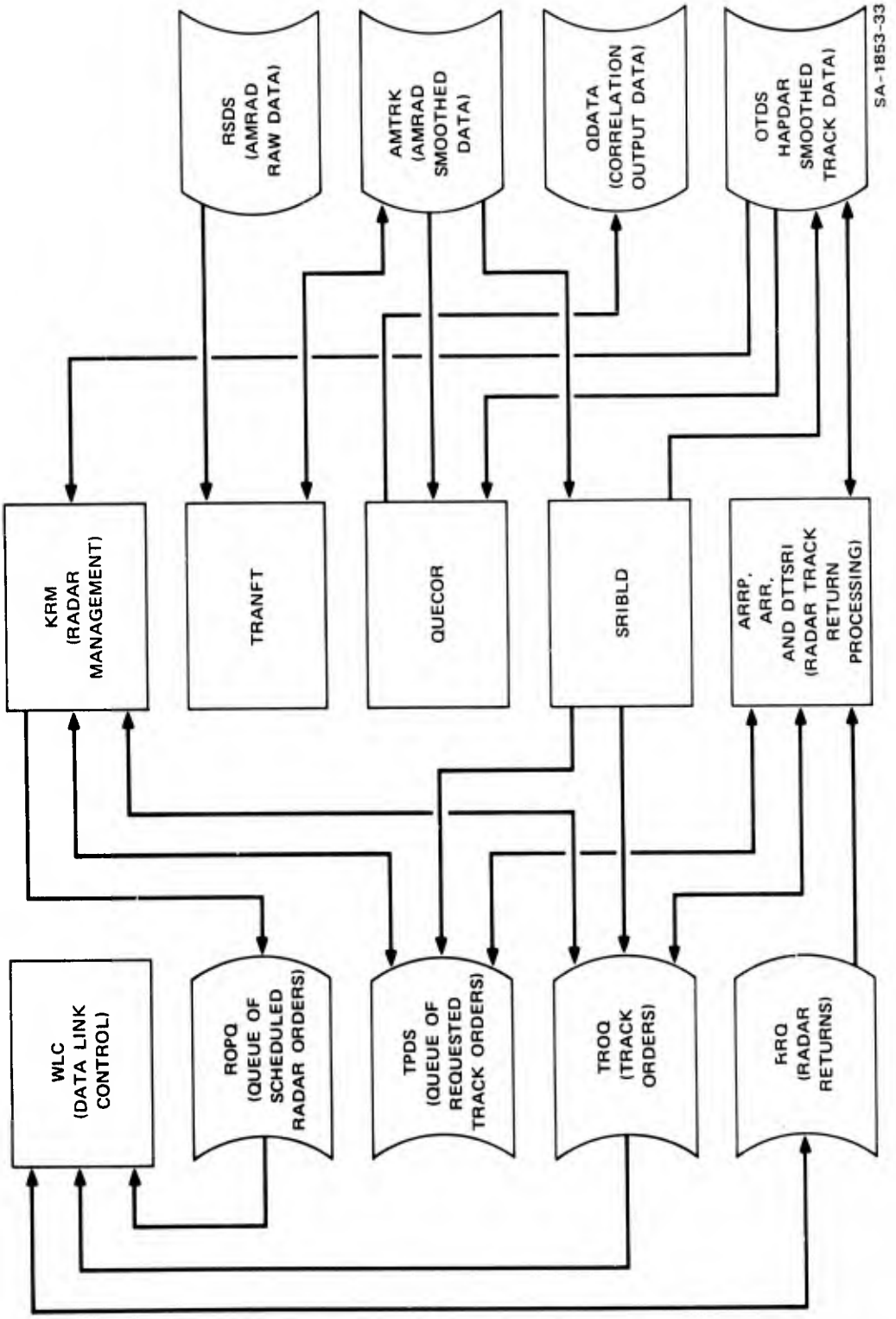


FIGURE E-1 PHSD/SRI HANDOVER SOFTWARE INTERFACE DIAGRAM

come in to RSDS and TRANFT at a rate of 20 per second. This corresponds to a data rate of 10 per second per AMRAD track. Thus, TRANFT is executed 20 times per second, but calls QUECOR and SRIBLD only once per second. At the times when TRANFT does not request a handover, control of the CPU is returned to BMDOS.

The handover process is initiated when TRANFT calls QUECOR. The function of QUECOR is to compute the value of the correlation parameter, Q, for all track pair associations consisting of the AMRAD track to be handed over and each of the HAPDAR active tracks. A threshold criterion is applied to Q to determine whether the AMRAD track is correlated with any of the HAPDAR tracks, and if so, with which one. In a tactical situation a decision might be made at this point whether or not to continue the handover process. For the field test version of QUECOR, the handover is attempted regardless of the results of QUECOR data processing.

In order to compute each value of Q, QUECOR obtains the filtered track data for the AMRAD track to be handed over from AMTRK, and the filtered track data for each HAPDAR track in the OTDS data file. The OTDS data file is a variable length or volatile file that contains the track data for all of the HAPDAR active tracks. This includes two special dedicated tracks that can never be purged once they are built. These two dedicated tracks are employed in the SRI handover experiments. Each instance in the OTDS corresponds to a track channel. The instances contain the state vector, the covariance matrix, the time of validity, and other information on each track. The results of the correlation processing of QUECOR are loaded into a data file called QDATA. After QUECOR processing is completed, control is returned to TRANFT which immediately calls SRIBLD.

The functions of SRIBLD are to initially build the two dedicated track instances in the OTDS, reinitialize the appropriate dedicated OTDS instance with data from one of the AMRAD tracks, and order a track pulse transmission for that track. Thus, when SRIBLD is called it obtains the filtered data from AMTRK corresponding to the track to be handed over. If this is the first time SRIBLD has been called to process this particular AMRAD track, it builds a dedicated OTDS instance and initializes it with data obtained from AMTRK. Otherwise, it simply reinitializes the proper dedicated OTDS instance with the AMTRK data. SRIBLD then requests a time for transmission of a track pulse by placing a TPDS instance in an active queue with the desired time. The TPDS data file is a volatile file, each of whose instances correspond to one of the active HAPDAR tracks. The instance contains a track pulse transmission time, a pointer or address of the OTDS instance, and a pointer to the corresponding instance of the TROQ data file. The TROQ data file contains the data required to form the track beam being requested through TPDS.

In order to ensure a high priority for this handover track pulse request, the request time is set equal to the time of validity of the data in AMTRK. This time will generally be slightly in the past, but will be honored in any case as soon as possible. SRIBLD also constructs part of the track order contained in the TROQ. This consists of setting some pointers in TROQ, setting the track/special acquisition indicator to "track," the type of command to "SRI track," and the waveform to "long range track." SRIBLD then returns control to TRANFT, which then returns control to BMDOS.

At an appropriate time the radar management function, KRM, performs two services. First, it selects and schedules radar pulses; second, it builds radar orders for each of the pulses. These radar orders may be for either track, verify, or search pulses. Only the track pulse scheduling function interfaces with the SRI handover software.

KRM examines the TPDS data file to determine if any track requests are present. If so, KRM selects the instance in TPDS with the earliest time such that the time does not exceed the block end time for the next block of radar orders to be scheduled. If there is a valid choice, then KRM detaches the instance from the TPDS queue (i.e., flags it to indicate it has been scheduled). KRM then completes the corresponding radar order instance in TROQ. This includes computation of such quantities as beam-pointing information and estimated range, among other associated information. In order to perform these computations, data are obtained from the corresponding instance of OTDS. KRM then puts the pointer to the TROQ instance it has just completed into a radar order pointer queue, ROPQ. ROPQ is a data file consisting simply of a list of such pointers for the radar orders scheduled.

KRM continues reiterating this procedure until a radar order block is filled, or TPDS is empty or an ineligible request is encountered (i.e., one with a request time later than the block end time). KRM then proceeds to examine the verify and search requests, if the block has not been filled.

After completing a radar order block, KRM checks the required radar duty factor for the scheduled block. If the radar duty factor specification is exceeded, KRM simply increases the block duration. At this point KRM returns control to the BMDOS system.

After KRM has been executed, the data link control function, WLC, is executed. WLC employs a set of routines to transmit radar orders through the radar computer interface (RCI) to HAPDAR and receive radar returns generated by HAPDAR. The orders transmitted over the data link are passed via the set of pointers in ROPQ. The data link software then obtains the data from TROQ (when a track order is to be transmitted) and converts the radar orders to the form required by the HAPDAR data

link. Similarly, the data of the returns from HAPDAR are converted to the proper form and placed in a unique instance in the radar return queue, RRQ. RRQ is a data file that contains the raw data obtained by HAPDAR. When the data link software completes the conversion of the set of returns, the RRQ data file is passed to the Return Preprocessing function ARRP.

ARRP processes the returns that have been established in RRQ. It first determines the type of return (i.e., search, verify, special acquisition, or track return) and then calls one of several appropriate return processing functions. In particular, if the return is a hand-over return, the function DTTSRI is called.

DTTSRI and its associated subfunctions filter the HAPDAR data obtained in response to track pulse requests for the two dedicated track channels. If a valid hit is not obtained, DTTSRI simply propagates the state and covariance up to the current time. If there is a valid hit, the raw data contained in RRQ in conjunction with information contained in TROQ is smoothed into the data in OTDS, and the new smoothed data is placed into OTDS. In addition DTTSRI requests the next track pulse by placing an appropriate TPDS instance into the active queue. The processing completes a normal cycle when KRM is executed again to build and schedule the next radar orders. Once every second, as discussed above, this normal cycle is reinitialized when TRANFT calls QUECOR and SRIBLD.

Appendix F

DESCRIPTION OF AMRAD MODIFICATIONS FOR TARGET HANDOVER AND CORRELATION DEMONSTRATION

Appendix F

DESCRIPTION OF AMRAD MODIFICATIONS FOR TARGET HANDOVER AND CORRELATION DEMONSTRATION

The Riverside Research Institute (RRI) made major modifications to the AMRAD equipment to support the handover and correlation program. These modifications were described in a proposal submitted to WSMR.[12] The following is an updated abstract from the proposal, describing the handover system, the data transmission system, and the synchronization and timing of the systems.

A. System Description

The equipment configuration to provide the Phase II* AMRAD/HAPDAR interface is shown in Figure F-1. The implementation of the Phase II configuration primarily entailed (1) providing an additional data path between DRTS II DAS and the AMRAD/HAPDAR interface system, (2) modification of the SDS 920 software to eliminate the prediction algorithms and to add the Range Tracker II data handling capability, and (3) redesign of the kineplex data format.

The additional data paths required that signal conditioning and cable driving circuitry be added in the DAS II for 22 bits of range information, 7 bits of precursor A/D converter counts, and 3 bits of track status information. The MILGO chassis formerly used as the chain storage register was completely redesigned to accept and store the 22 bits of secondary range data. The MILGO phase and amplitude

---

* Phase I is described in Reference 2.

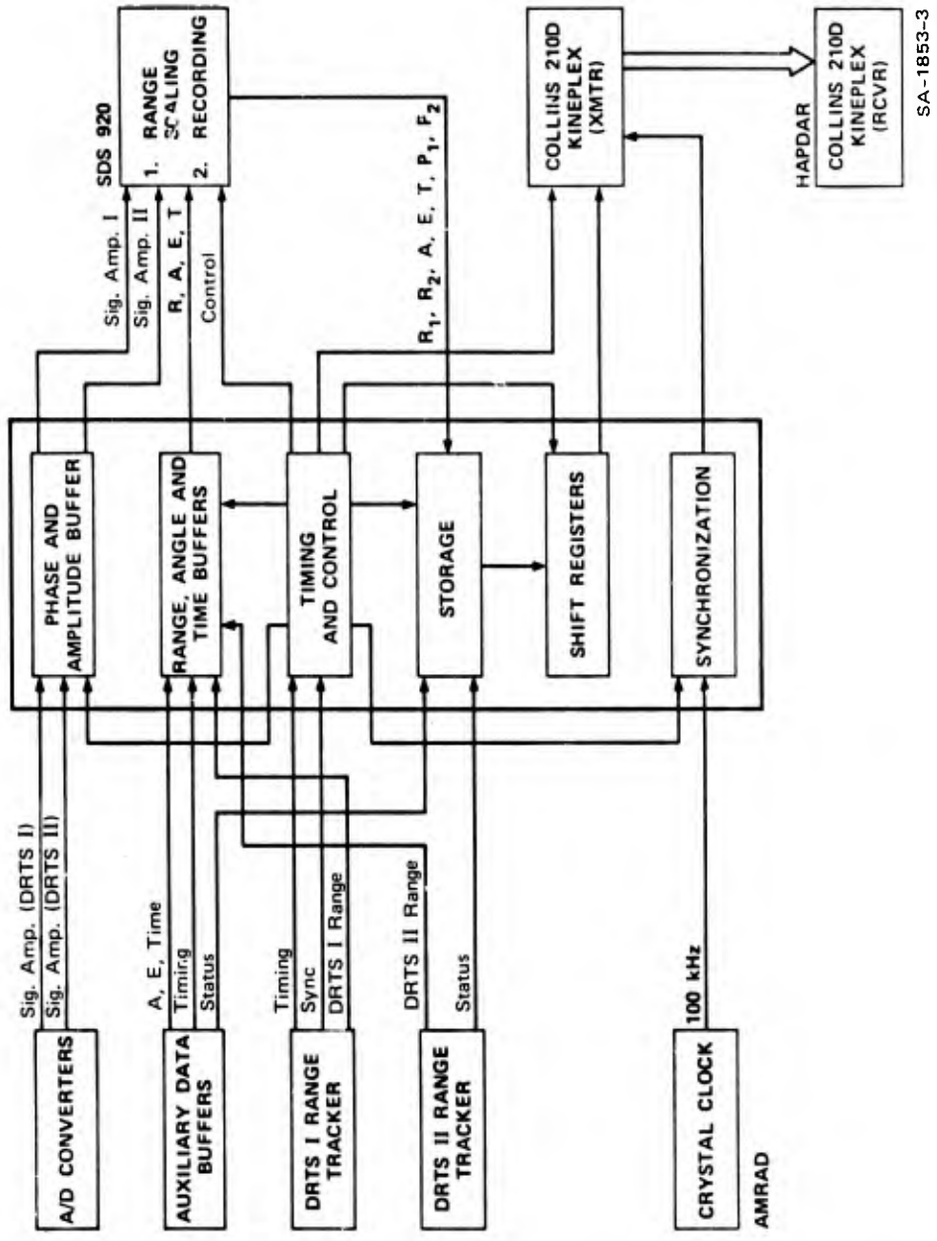


FIGURE F-1 PHASE II CONFIGURATION (AMRAD/HAPDAR INTERFACE)

buffer was modified to accept and store the seven bits of precursor A/D counts. The transmitter control chassis was modified to accept and gate the track status information for both trackers.

The prediction algorithm was eliminated, and logic to store, record, and output raw secondary tracker data was provided. Signal amplitude data are averaged for each tracker and output to the kineplex in a format suitable for setting AGC levels at HAPDAR. Raw range data are output to the kineplex in a 20-bit binary format.

The kineplex data format was a slightly modified version of the Phase I formats. The format for each tracker was identical. The specific tracker with which the data are associated was identified by a code in channel 4 of frame 1. Sync identification transmitted in frame 13 of the Phase I format was replaced by signal level information. The time-of-day at which the data was valid was contained in frames 5 and 9.

The implementation of format changes involved in frames 1 through 12 required only minor software modifications. However, since frame 13 was under hardware control, a new addressable holding register and gating control was required. A gating function was designed to allow either the hardware controlled sync identification or the software controlled signal strength information to be manually selectable, so as to provide flexibility during the development of the Phase II interface.

B. Kineplex Data Description

The kineplex message format was identical for both DRTS I and DRTS II tracker information and is shown in Figure F-2. Definitions of items in the data are tabulated below.

Item	Definition
Tracker identification (I_1, I_2)	Tracker identification is contained in channels 3 and 4 of frame 1. I_1 = 1 = AMRAD I_1 = 0 = PAS I_2 = 1 = DRTS II tracker I_2 = 0 = DRTS I tracker
Range (R_1 - R_{20})	Range information is contained in frames 1 through 4. R_1 = MSB = 1.4989644×2^{19} meters R_{20} = LSB = 1.4989644×2^{0} meters
Azimuth (A_1 - A_{17})	Azimuth information is contained in frames 6 through 8. $A_1 = 6400 \times 2^{-1}$ mils (180°) $A_{17} = 6400 \times 2^{-17}$ mils ($180° \times 2^{-16}$)
Elevation (E_1 - E_{17})	Elevation information is contained in frames 10 through 12. $E_1 = 6400 \times 2^{-1}$ mils (180°) $E_{17} = 6400 \times 2^{-17}$ mils ($180° \times 2^{-16}$)
Time (T_1 - T_{12})	Time of day at which data was valid (referenced in GMT and contained in frames 5 and 9). T_1 = 2 seconds T_2 = 1 second $T_3 = 1 \times 2^{9}$ milliseconds (512 ms) $T_{12} = 1 \times 2^{0}$ milliseconds T_3 through T_{12} contain a binary count of milliseconds that truncates to zero after a count of 999 milliseconds.

Received power ($P_1 - P_6$)

Average received power, relative to AMRAD received precursor, is contained in frame 13. $P_1 - P_6$ is the binary magnitude of AMRAD received power (in dBm) above the predefined HAPDAR minimum discernible signal where:

P_1 = MSB = 1.0×2^5 (dBm)

P_6 = LSB = 1.0×2^0 (dBm)

Tracker status (S_1, S_2)

The status of the tracker identified in frame 1 is contained in frame 14. The valid codes are as follows:

S_1	S_2	
0	0	manual
0	1	acquisition
1	1	autotrack

Frame 15 was hardwired to contain all zeros. Channels 7 and 8 contain identical information, which is all ones.

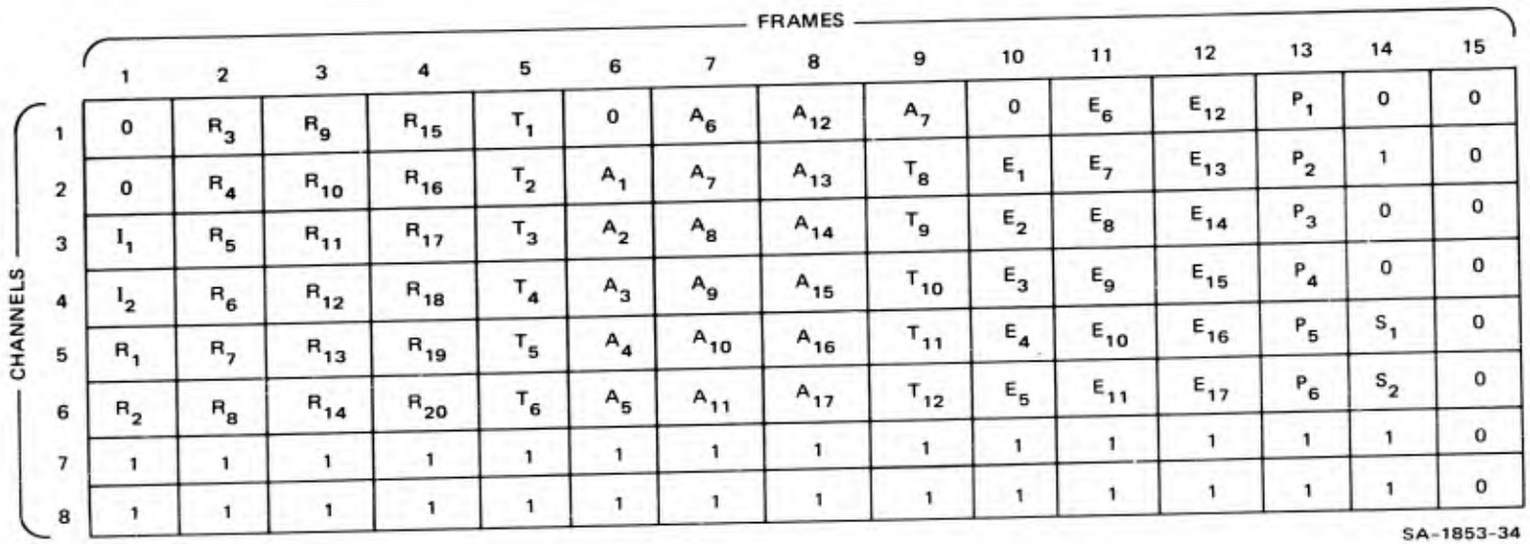


FIGURE F-2 PHASE II AMRAD/HAPDAR KINEPLEX FORMAT

REFERENCES

1. J. Naar and P. R. Albee, "Radar Netting Studies Phase II (U)" SRI Project 8876, Final Technical Report, Contract F30602-71-C-0045, Stanford Research Institute, Menlo Park, California (October 1971), SECRET.

2. E. T. Brandon, D. G. Falconer, and H. A. Olender, "Radar Netting Studies Phase III (U)" SRI Project 1473, Final Technical Report, Contract F30602-72-C-0060, Stanford Research Institute, Menlo Park, California (July 1972), SECRET.

3. K. J. Kahrilas, "HAPDAR-An Operational Phased Array Radar," Proceedings of the IEEE, Vol. 56, No. 11, pp. 1967-1975 (November 1968).

4. T. H. Witzel and J. J. Selfridge, "Preliminary BMD Software Develment for IBM DP Systems" Volumes I through IX, Final Report, Contract DAHC60-70-C-0052, International Business Machines Corporation, Gaithersburg, Maryland (18 January 1972).

5. Ben H. Cantrell, "Behavior of α-β Tracker for Maneuvering Targets under Noise, False Target, and Fade Conditions," NRL Report 7434, Naval Research Laboratory, Washington, D.C. (17 August 1972).

6. E. T. Brandon, "Test Plan for Field Demonstrations for the Sensor Netting Program," Technical Note SED-TN-559, Stanford Research Institute, Menlo Park, California (19 September 1972).

7. R. H. Kind, "AMRAD Data Summary for Sounding Rocket SRI-04/Rome-03 Flight Flown on 28 February 1972," Research Note N-48/241-4-10, Riverside Research Institute/AMRAD, El Paso, Texas (9 March 1973).

8. D. G. Falconer and H. A. Olender, "Software Specifications for Sensor Netting Program," Technical Note SED-TN-557, Stanford Research Institute, Menlo Park, California (14 August 1972).

9. R. A. Fisher and F. Yates, Statistical Tables for Biological, Agricultural, and Medical Research (Oliver and Boyd, Edinburgh, 1949).

10. M. Abramowitz and I. A. Stegun, ed., Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables, pp. 942-943, National Bureau of Standards Applied Mathematics Series 55 (U.S. Covernment Printing Office, Washington, D.C., 1964).

11. "HAPDAR BMD Software System, Software Design Specification, HAPDAR BMD Tactical Process," Technical Report, Contract DAHC60-72-G0032, RCA, Moorestown, New Jersey (July 1972).

12. "Proposal for AMRAD Major Modifications in Support of Follow-on Phase of Target Handover and Correlation Demonstration at WSMR," Riverside Research Institute, AMRAD Facility, El Paso, Texas (13 July 1972).

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified

1 ORIGINATING ACTIVITY (Corporate author)
Stanford Research Institute
333 Ravenswood Ave., Menlo Park, Calif. 94025

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3 REPORT TITLE
SENSOR NETTING PROGRAM Volume II: Field Demonstrations

4 DESCRIPTIVE NOTES (Type of report and inclusive dates)
Final Report

5 AUTHOR(S) (First name, middle initial, last name)
Earl T. Brandon Henry A. Olender David G. Falconer

6 REPORT DATE	7a. TOTAL NO OF PAGES	7b. NO OF REFS
May 1973	204	12

8a. CONTRACT OR GRANT NO	9a. ORIGINATOR'S REPORT NUMBER(S)
DAHC60-70-C-0016	SRI Project 1853
b. PROJECT NO	Final Report
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10 DISTRIBUTION STATEMENT

11 SUPPLEMENTARY NOTES	12 SPONSORING MILITARY ACTIVITY
	U.S. Army Advance Ballistic Missile Defense Agency Attn: RDMH-C, P. O. Box 1500, Huntsville, Alabama 135807

13 ABSTRACT

For sensor netting in ballistic missile defense (BMD), data must be transferred from one sensor to another. The purpose of the field demonstrations reported here was to demonstrate specific netting functions entailing data transfer from one radar to another remotely located radar under field conditions. At the White Sands Missile Range, target data from various objects were transferred from AMRAD, a mechanical tracking radar, to HAPDAR, a phased array radar driven by an IBM 360/65 computer using special BMD software. The two radars were separated by about ten miles.

This report describes the software developed for these demonstrations and the tests that were carried out to demonstrate target transfer from one radar to another and target track association or correlation between independent tracks of the two radars.

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
radar netting						
ballistic missile defense						
target correlation						
target handover						
search handover						